Universidade Federal da Bahia
Instituto de Matemática / Escola Politécnica

Programa de Pós-Graduação em Mecatrônica

GAZE ESTIMATION VIA ATTENTION-AUGMENTED CONVOLUTIONAL NETWORKS

Gabriel Lefundes Vieira

DISSERTAÇÃO DE MESTRADO

Salvador
25 de novembro de 2020

GABRIEL LEFUNDES VIEIRA

GAZE ESTIMATION VIA ATTENTION-AUGMENTED CONVOLUTIONAL NETWORKS

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Mecatrônica da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Mecatrônica.

Orientador: Luciano Rebouças de Oliveira

Salvador
25 de novembro de 2020

V658 Vieira, Gabriel Lefundes.
Gaze estimation via attention-augmented convolutional networks / Gabriel Lefundes Vieira. – Salvador, 2020.
97 f.: il. color.

Orientador: Prof. Dr. Luciano Rebouças de Oliveira.

Dissertação (mestrado) – Universidade Federal da Bahia. Escola Politécnica, 2020.

1. Olhar - estimativa. 2. Visão computacional. 3. Redes neurais. I. Oliveira, Luciano Rebouças de. II. Universidade Federal da Bahia. III. Título.

CDD: 006.3

Termo de Aprovação

GABRIEL LEFUNDES VIEIRA

Gaze estimation via attention-augmented convolutional networks

Esta Dissertação de Mestrado foi julgada adequada à obtenção do título de Mestre em Mecatrônica e aprovada em sua forma final pelo Programa de Pós-Graduação em Mecatrônica da Universidade Federal da Bahia.

Salvador, 25 de novembro de 2020

Prof. Dr. Luciano Rebouças de Oliveira (UFBA)

Prof. Dr. Eduardo Simas (UFBA)

Prof. Dr. Carlos Hitoshi Morimoto (USP)

ACKNOWLEDGEMENTS

First, I'd like to acknowledge my advisor Prof. Dr. Luciano Rebouças de Oliveira for the opportunity to conduct this research and for the invaluable guidance in both academia and life that he has given me in the past two years. He inspired me to pursue computer vision and AI as a professional field, and I can't imagine myself doing anything else now. I'd also like to thank all my colleagues from the Intelligent Vision Research Lab for the camaraderie during my time there. I learned more than I ever expected over these past years and this is in large part thanks to them.

I'd like to acknowledge all the professors, staff, and fellow researchers of the CTAI-UFBA and Escola Politécnica-UFBA for the teachings, support, and companionship during this time.

The work I have done throughout my Master's would not have been possible were it not for the Universidade Federal da Bahia, which has been my second home for a considerable part of my life. Each year spent as a part of its community has changed me for the better. Given the current political landscape of our country, it is paramount to support, defend, and recognize the value that free, accessible, and quality education provided by institutions such as UFBA bring to our society. We are still far from being ideal, but attacks on the Brazilian educational system and other public social institutions cannot be tolerated. On this note, I acknowledge the financial support from the Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB) [grant number 892/2019], and the National Laboratory for Scientific Computing (LNCC/MCTI, Brazil) for providing HPC resources of the SDumont supercomputer (URL: http://sdumont.lncc.br) without which this research would also not have been possible.

I'd like to thank my friends, old and new, who have mercilessly questioned my life choices every step of the way giving me the clarity of mind to make the (hopefully) right decisions for my professional and personal lives. I will not be naming their names for the sake of brevity, and also due to the inevitable fact that I will forget someone whose feelings I do not wish to hurt, and I have no interest in dealing with this gaffe.

I want to thank my partner, Charmila, for being the most supportive and caring person anyone could ever ask for. She has helped me hold on to my motivation and energy to carry on with my ambitions when, in multiple instances, it felt like an easier alternative to just not.

I would also like to acknowledge and thank my family profusely. In particular, my

sister Thais, my father Aldo, and my mother Ana Rita. They are the ones I think about when picturing what a good, kind, competent, and responsible person should be. They have supported me unconditionally throughout my life, and without this support, I can unequivocally say I would not be where I am today, so thank you.

Finally, I'd like to thank my therapist for allowing me to vent and complain about every single other person mentioned in this section so far.

A minha alucinação é suportar o dia-a-dia, e meu delírio é a experiência com coisas reais

—BELCHIOR (Alucinação)

RESUMO

A estimativa de olhar (do inglês — *gaze estimation*) é altamente relevante para aplicações em vários campos, incluindo, mas não se limitando a, sistemas interativos, interfaces homem-computador e pesquisa comportamental. Como muitas outras tarefas de visão computacional, a estimativa do olhar se beneficiou muito com o avanço do aprendizado profundo (do inglês — *deep learning*) na última década. Recentemente, uma série de *data sets* de larga escala para estimativa de olhar baseada em aparência foram tornados públicos, e redes neurais foram estabelecidas como a abordagem padrão do estado-da-arte para essa tarefa. Atualmente, porém, ainda há espaço para melhorias no que diz respeito às arquiteturas de rede usadas para realizar a estimativa do olhar com base na aparência.

Um caminho promissor para melhorar a precisão da estimativa do olhar é levar em consideração as informações de pose da cabeça que são implicitamente presentes em imagens faciais. Alguns trabalhos conseguem aproveitar essa informação usando a imagem inteira do rosto como entrada para a rede neural. Uma desvantagem dessa estratégia é que as redes neurais convolucionais tradicionais não são capazes de formar relações espaciais entre elementos distantes de uma imagem. Este é um fator significativo na estimativa da pose da cabeça, dado que a mesma é determinada por uma combinação de diferentes características de olhos, nariz, boca, etc. Tendo isso em mente, aqui propomos uma nova abordagem que usa camadas de convolução melhoradas por auto-atenção para aumentar a qualidade das caracteristicas aprendidas pela rede, dando à mesma a capacidade de formar relações espaciais complexas de longo alcance. Nossa abordagem, quando aplicada a redes residuais mais rasas, pode ajudá-las a superar arquiteturas profundas, aprendendo a identificar dependências entre regiões distantes em imagens de rosto inteiro, criando representações mais sensíveis ao espaço a partir das imagens de rosto e olhos. Essa representação é utilizada para estimação do olhar através de uma regressão. Um efeito colateral interessante de nossa abordagem é também que ela é capaz de criar representações intermediárias mais interpretáveis visualmente, derivadas dos pesos usados pelas camadas de auto-atenção, possivelmente permitindo discussões interessantes sobre o processo de aprendizagem da rede.

Denominamos nosso framework de estimativa de olhar como **ARes-gaze**. Ele explora nossa **A**tention-augmented **Res**Net (**ARes-14**) como backbones convolucionais gêmeos para processamento das entradas. Em nossos experimentos, os resultados mostraram uma diminuição do erro angular médio em 2,38% quando comparados aos métodos mais modernos no *data set* MPIIFaceGaze e alcançaram o segundo lugar no *data set* EyeDiap. É importante notar que nosso framework proposto foi o único a atingir alta precisão simultaneamente em ambos os conjuntos de dados entre os métodos avaliados.

Palavras-chave: Estimação de olhar, deep learning, visão computacional

ABSTRACT

Gaze estimation is highly relevant to applications in multiple fields, including but not limited to interactive systems, specialized human-computer interfaces, and behavioral research. Like many other computer vision tasks, gaze estimation greatly benefited from the advancement of deep learning in the past decade. A number of large scale data sets for appearance-based gaze estimation have been made public, and neural networks have been established as the core for the state-of-the-art approaches to this task. Currently, there is still room for improvement with regards to the network architectures used to perform appearance-based gaze estimation.

One promising avenue to improve gaze estimation accuracy is to take into account the head pose information contained in facial images. A few published works do this by using the entire face image as the input to the network. One drawback to this strategy is that traditional convolutional neural networks are not able to form long-range spatial relationships within images. This is a significant factor in head pose estimation, given that the pose is determined by a combination of different features from eyes, nose, mouth, etc. To this effect, here we propose a novel approach that uses self-attention augmented convolution layers to improve the quality of the learned features. This is done by giving the CNN the ability to form long-range complex spatial relationships. We propose the use of a shallower residual network with attention-augmented convolutions, which we dubbed **ARes-14**. We show that by using Ares-14 as a backbone, it is possible to outperform deeper architectures by learning dependencies between distant regions in full-face images, creating better and more spatially-aware feature representations derived from the face and eye images before gaze regression. An interesting side-effect of our approach is also that it can create more visually-interpretable intermediary representations derived from the attention weights used by the self-attention layers, enabling interesting discussions about the learning process of the network.

We dubbed our gaze estimation framework as **ARes-gaze**, which explores our **A**ttention-augmented **Res**Net (**ARes-14**) as twin convolutional backbones. In our experiments, results showed a decrease of the average angular error by 2.38% when compared to state-of-the-art methods on the MPIIFaceGaze data set, and achieved second-place on the EyeDiap data set. It is worth noting that our proposed framework was the only one to reach high accuracy simultaneously on both data sets among the evaluated methods.

Keywords: Gaze estimation, deep learning, computer vision

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ML	Machine Learning	8
DL	Deep Learning	
LOS	line-of-sight	2
CNNs	convolutional neural networks	8
RNN	recurrent neural network	12
AAConv	attention-augmented convolution	29
FC	fully connected	xvi
SE	Squeeze-and-Excitation	27
BAM	Bottleneck Attention Module	28
CBAM	Convolutional Block Attention Module	28
LSTM	Long short-term memory	13
GRU	Gated Recurrent Unit	13
FLOPs	floating point operations	36
SEI	Stacked-eyes input	48
DP-SW	Double Pass - Shared Weights	48
DP-TB	Double Pass - Twin Branches	48

CHAPTER 1

INTRODUÇÃO

Contents

Gaze estimation is an active area of research within computer vision, and its relevance spans a wide array of fields. For instance, gaze can be a valuable source of information in behavioral and health research (HOLLANDS; PATLA; VICKERS, 2002; WARLOP *et al.*, 2020; NAKANO *et al.*, 2010), augmented and virtual reality (NILSSON; GUSTAFSSON; CARLEBERG, 2009; LANKES; STIGLBAUER, 2016; LEE *et al.*, 2011), mobile applications (BARZ *et al.*, 2018; LANKES; STIGLBAUER, 2016), and even natural language processing (NLP) (HAKKANI-TÜR *et al.*, 2014). As more and more applications in these areas become available in the mainstream, the relevance of gaze as an input modality also increases.

Gaze estimation methods can be categorized as model-based or appearance-based (HANSEN; JI, 2009). Model-based systems explicitly model the geometric structures of the human eye in a simplified manner, usually by assuming spherical shapes for the eyeball and the cornea (HANSEN; JI, 2009). Model-based approaches rely heavily on controlled and well-calibrated environments. In general, to accurately calculate the 3D gaze direction vector from images and geometrical models, one needs previous knowledge of the subject's distance to the cameras, a fixed-position light source (either an environmental light fixed with regards to the subject's position or an active component such as infrared LEDs placed near the cameras (ZHU; JI, 2004)), and complex camera setups (sometimes using multiple cameras (ZHU; JI; BENNETT, 2006), stereo cameras (OHNO; MUKAWA, 2004), mirrors (NOUREDDIN; LAWRENCE; MAN, 2005), or cameras mounted on moving pan-tilt units (TALMI; LIU, 1999)) to account for head movement.

As accurate as these model-based methods may be, their complicated nature makes it very hard to deploy them to scale and in unconstrained environments. In contrast, appearance-based gaze estimation methods do not need camera calibration or geometric modeling of eye and environment. This class of methods uses image data directly as inputs to a mapping function that outputs 2D screen coordinates or 3D gaze direction vector angles

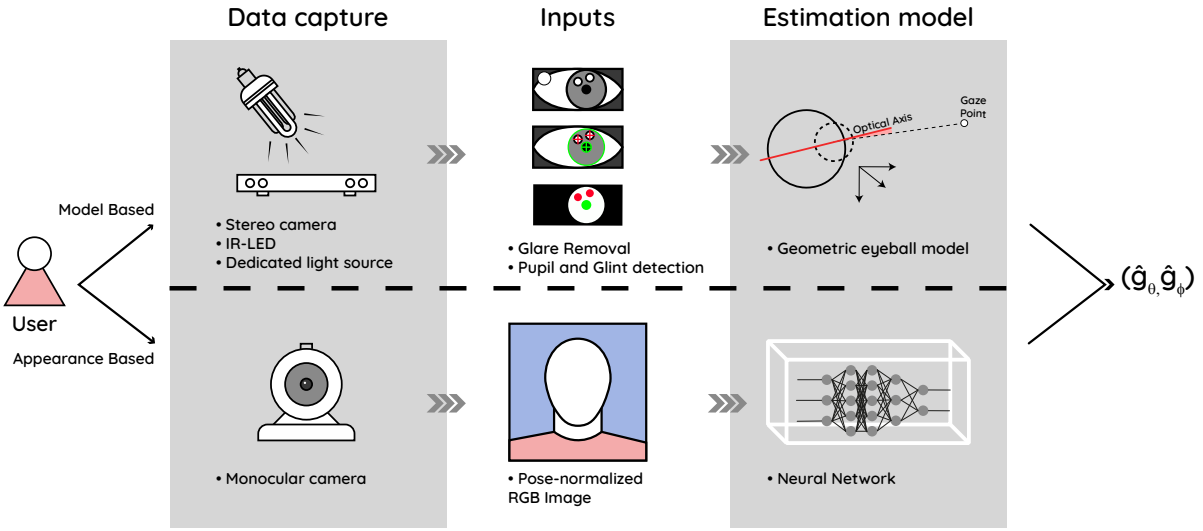


Figure 1.1: Comparison of model- and appearance-based gaze estimation pipelines. The branch performed by the appearance-based model is less complicated in every step, requiring less hardware for the data capture, less pre-processing, and with the result being obtained by performing a simple forward pass through the prediction model.

Figure 1.1 shows a comparison between the equivalent steps of a sample model-based algorithm and a sample appearance-based one. The top row shows a simplified model-based approach which uses a stereo camera to capture images and estimate the subject's distance in conjunction with IR-LEDs used to produce glint points in the subject's pupils. Regular image processing is used to detect and segment the regions of the iris, pupils, and reflected glints. These features can be mapped to a mathematical model of the eyeball to produce a line-of-sight (LOS) gaze vector. This kind of algorithm is common in commercial eye-tracking software for computer screens, where the final result is the point-of-gaze (pixel) where the user is focusing on at a given moment in time. This point-of-gaze is obtained by calculating the geometric intersection between the LOS vector and the camera's plane.

The bottom row of Fig. 1.1 shows a sample appearance-based model. A monocular camera is used to obtain simple RGB images of the subject. These images are normalized and fed to a learned mapping function that predicts the pitch and yaw angles of the gaze vector. In this case, the mapping function is a neural network. This pipeline is a rough simplification of the gaze estimation framework proposed by this document.

A lot of different algorithms have been proposed as mapping functions in appearance-based gaze estimation such as manifold learning (TAN; KRIEGMAN; AHUJA, 2002), Gaussian processes (WILLIAMS; BLAKE; CIPOLLA, 2006) and learned non-linear regression (MARTINEZ; CARBONE; PISSALOUX, 2012). The main limitation of this approach is that eye images can look the same while looking at different directions due to changes in head pose. Additionally, since illumination is not handled explicitly, two pictures of an eye looking in the same direction can look different according to the lighting conditions of the environment at the moment. More recently, with the great success

achieved by deep convolutional neural networks in computer vision and the publishing of large-scale data sets (ZHANG *et al.*, 2015; FISCHER; CHANG; DEMIRIS, 2018; SUGANO; MATSUSHITA; SATO, 2014; SMITH *et al.*, 2013; KRAFKA *et al.*, 2016), these problems are being overcome by using neural networks as mapping functions and training on large volumes of data. A typical modern appearance-based gaze estimation pipeline has the following steps:

- **Image acquisition and face detection:** A face-detection algorithm estimates the subject's position by analyzing images obtained from a simple monocular camera. Some systems use the entire detected face region as input (ZHANG *et al.*, 2017; KRAFKA *et al.*, 2016; CHEN; SHI, 2020), some use the eye regions, some use of both. The face is included to encode the head pose information implicitly. If the method in question takes as input the eye regions, a face-landmark detector can be used to locate eye coordinates from within the face image.

- **Input pre-processing:** The input data (face, eyes) needs to be normalized to resemble the data used to train the network. This process usually involves resizing the image to reference height and width values. Some channel-wise preprocessing can also be applied such as histogram normalization (if the images are in grayscale), to reduce the impact of different lighting conditions. For face images specifically, it's also common to apply an affine transform as to cancel-out the roll-axis angle of the face to reduce complexity.

- **Prediction:** The normalized data is fed to the network. There's no need for geometrical calculations as it might be necessary for model-based approaches. The output is a pair of values representing the yaw and pitch angles that can be used to reconstruct the estimated line-of-sight of the subject. The scope of this work is mainly constrained to this step.

- **Gaze projection (optional):** If the application's final objective is to locate screen coordinates to where the subject's gaze is directed at, it needs additional information to translate the line-of-sight (LOS) into usable data. A traditional approach would be to obtain the camera's calibration matrix and geometrically calculate the intersection point between the gaze line and the plane representing the screen. It is possible to incorporate this step into the prediction stage by training the network to output screen coordinates directly as done in (KRAFKA *et al.*, 2016), where the detected face's location and scale are fed to the neural network to aid in this task. The downside of this approach is that the trained model is nearly useless outside of the relatively constrained training environment.

In this work, we explore the recent trend of attention mechanisms in deep learning (VASWANI *et al.*, 2017) as a way to produce higher quality features by improving the spatial awareness of the network. The use of full-face images along with the usually extracted eye-patches as the input has been shown to improve the prediction accuracy significantly (ZHANG *et al.*, 2017; KRAFKA *et al.*, 2016; CHEN; SHI, 2020) since full-face images carry relevant information about the subject's pose. Bare this in mind, we

attempt to further improve the use of that information by leveraging the spatial awareness afforded by the network via attention augmented convolutions.

1.1 MOTIVAÇÃO

The increasing demand for reliable gaze estimation methods for real-world applications merits the exploration of the rapidly evolving deep learning environment for computer vision. In the current landscape of relevant research, attention-augmented convolutions are particularly promising since their strong points (spatial awareness) match well with the gaze estimation domain.

In (BELLO *et al.*, 2019) attention-augmented convolutions are shown to improve accuracy for many different architectures on the baseline ImageNet (RUSSAKOVSKY *et al.*, 2015) data set for classification, with some experiments suggesting that the network can learn semantically relevant regions and features on the input image.

Two main questions drove the development of this work: *Can attention-augmented convolutions be used to improve the accuracy of appearance-based gaze estimation? How so?* and *Does the use of attention augmentation presents any clear advantages with regards to the traditional challenges of appearance-based gaze estimation (head pose and illumination)?*. This document seeks to present answers to these questions by experimenting with AAConvs in the domain of gaze estimation, proposing a fully augmented architecture.

1.2 OBJETIVOS

1.2.1 Objetivos gerais

The main goal of this work is to study the impact of attention-augmented convolutions when applied to the appearance-based gaze estimation task and propose a trainable gaze estimation framework with comparable or better performance to the current state-of-the-art by relying on attention augmentation.

1.2.2 Objetivos específicos

The specific goals of this work are as follows:

- Propose a CNN backbone deep enough to produce good results but shallow enough to compensate for the large computational overhead of attention-augmented convolution layers;
- Propose a trainable framework for appearance-based gaze estimation that performs comparably with the current state-of-the-art by applying the aforementioned attention-augmented backbone;
- Design experiments to evaluate the effects of attention augmentation in domain-specific challenges of gaze estimation, such as robustness to head pose and illumination conditions variation.

1.3 CONTRIBUIÇÕES

To the best of our knowledge, no prior work has explored attention-augmented convolutions in the gaze estimation task. To this effect, our specific contributions are:

- ARes-14, a ResNet-inspired attention-augmented network conceived for tasks that benefit from spatial awareness but do not require very deep architectures to be effective;
- The introduction of ARes-Gaze, a new appearance-based gaze estimation framework with attention-augmented backbones;

As a product of this work, the following paper has been submitted to a journal and is currently undergoing the review process:

Lefundes, G., & Oliveira, L. (2020). On estimating gaze by self-attention augmented convolutions. arXiv preprint arXiv:2008.11055.

1.4 MAPA DE CAPÍTULOS

The remainder of this Document is structured as follows:

- **Chapter 2** provides a brief introduction to the theoretical concepts of the appearance-based gaze estimation task and of attention mechanisms in deep learning, contextualizing the usefulness of the latter to the former;
- **Chapter 3** is dedicated to related works, outlining relevant and state-of-the-art appearance-based gaze estimation methods and works presenting attention-related mechanisms for deep neural networks;
- **Chapter 4** presents the core of our method by introducing ARes-14, our proposal of an attention-augmented feature extractor. In this chapter, we explain in details the concept of the architecture and the motivation for the design choices that resulted in its final version. Finally, we present the ARes-Gaze framework for gaze-estimation, addressing its data pipeline and overall model architecture;
- **Chapter 5** describes the methodology adopted to validate our proposal. In this chapter, we present evaluations on specific architectural aspects of our framework, comparing its performance against other similar methods on two popular gaze estimation data sets. Additional experiments are presented with regards to the impact of attention on robustness to head pose and illumination conditions;
- **Chapter 6** concludes the document by discussing the results presented in the previous chapter, putting them in context with the arguments made throughout the rest of the dissertation, and proposing future avenues of investigation.

CHAPTER 2

CONTEXTUALIZAÇÃO

Contents

In this chapter, we introduce the fundamental concepts used throughout the work. We show the theoretical formulation of the appearance-based gaze estimation task and how it relates in practice to modern approaches. We also briefly contextualize the history of attention mechanisms in deep learning, eventually explaining the inner-workings of the attention-augmented convolutional layer that is used in our proposal.

2.1 ESTIMAÇÃO DE OLHAR BASEADA EM APARÊNCIA

Appearance-based is a subset of gaze estimation methods. This section introduces the fundamentals necessary to understand how a solution to this problem is to be modeled.

Essentially, appearance-based gaze estimation techniques attempt to map an eye image directly to a gaze direction vector without explicit geometric formulations. The task itself can be defined as finding a function f capable of mapping an input image, $\boldsymbol{I}$, to a gaze vector, $\hat{\mathbf{g}}$. Given that a person's gaze direction and line-of-sight is usually also dependent on their head pose, ($\mathbf{h}$), we also include in the formulation, obtaining the traditional formula:

$$\hat{\mathbf{g}} = f(\mathbf{I}, \mathbf{h}) \,, \tag{2.1}$$

where $\hat{\mathbf{g}}$ is a 2D unit vector with the origin being in the middle point between the subject's eyes. The components that form $\hat{\mathbf{g}}$ are the pitch ($\hat{\mathbf{g}}_\theta$) and yaw ($\hat{\mathbf{g}}_\phi$) angles.

Over time, this concept extended to the use of entire facial images too, with the justification that this way, the input also contained head pose information that contributed

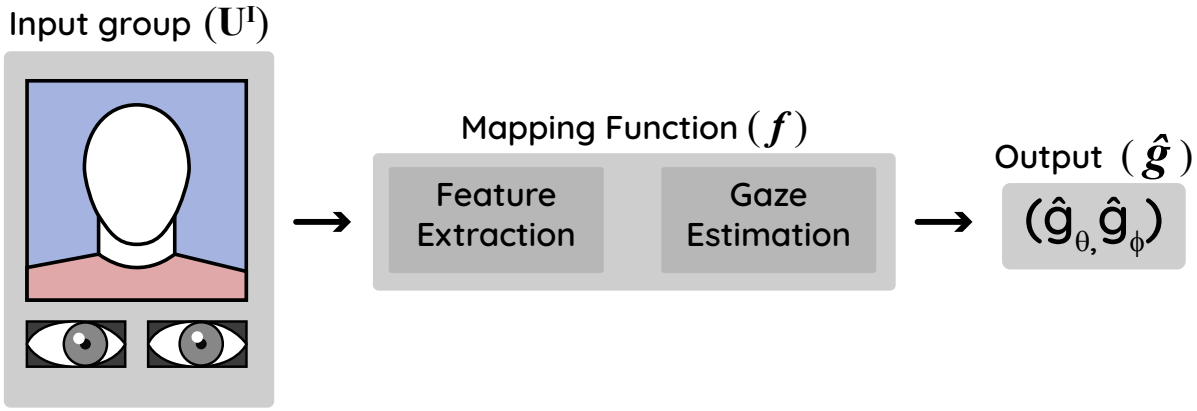


Figure 2.1: Simplified scheme for a sample appearance-based gaze estimation pipeline.

to the final prediction (ZHANG *et al.*, 2017). In these cases, $\mathbf{h}$ is implicitly inferred from the input image.

Many modern approaches go as far as to use a multi-modal input scheme, using both face and eye images, and sometimes other diverse information such as facial keypoints (PALMERO *et al.*, 2018) or face scale and position relative to the frame (KRAFKA *et al.*, 2016), as illustrated in Fig. 2.1. If we stipulate an input group $\mathbf{U}^I$ encompassing all possible input modalities, we can then extend the generic appearance-based formula to:

$$\hat{\mathbf{g}} = f(\mathbf{U}^I)\,. \tag{2.2}$$

2.2 DEEP LEARNING E REDES NEURAIS CONVOLUCIONAIS

The gaze estimation framework proposed here is equivalent to the mapping function f in the formulation presented in the previous section. In this document, the main focus will be on the use of deep learning and, more specifically, convolutional neural networks (CNNs) to serve as the f function. Theoretically, however, any learned function can play its role, with many traditional Machine Learning (ML) algorithms already proposed in the past as feature extractors for appearance-based gaze estimation. In this section, we attempt to answer what are the main differences between using traditional ML algorithms and CNNs in gaze estimation and explain the reasons to choose the latter over the former.

Machine Learning is a discipline inside the field of Artificial Intelligence, which focuses on the application of statistics, computer science, and neuroscience towards the goal of designing algorithms capable of learning from experience without explicit programming.

Deep Learning is a particular topic inside of ML in which artificial neural networks are used to learn how to extract features and create high-level representations of data in a more independent manner. In other words, given a sufficient amount of data, Deep Learning can learn relationships and structures without strict supervision in the lower levels of data representation.

Methods from what we call "classic" machine learning (support vector machines,

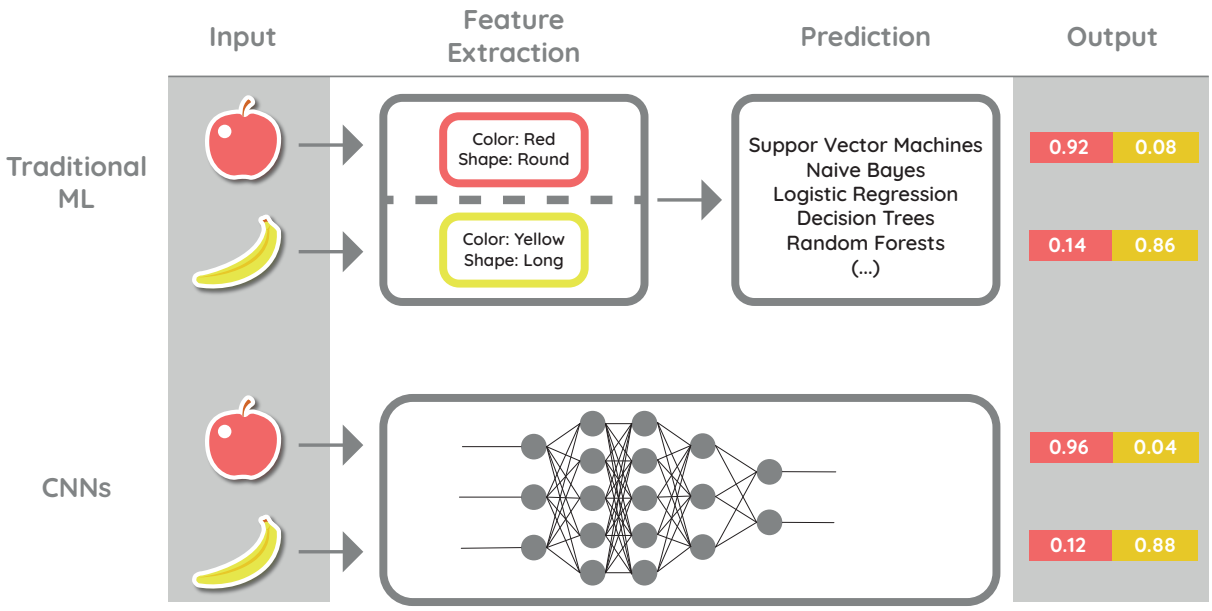


Figure 2.2: Conceptual illustration highlighting the differences between a traditional machine learning pipeline and a deep learning pipeline on the task of image classification. On the top row, traditional ML algorithms require that the features extracted from the input be modeled in some previous way, illustrated in the image as the "color" and "shape" attributes. On the bottom row, we see that neural networks perform the feature extraction and prediction stages jointly and implicitly.

random forests, etc) usually rely on a human expert that can interpret the data, organizing and explicitly codifying it into a set of features that will feed the learning algorithm. Deep learning methods have the purpose of reducing the role of the human expert in this equation by implicitly learning hierarchical patterns in the data. These patterns are implicit features, and the network learns to map them to the desired output. Figure 2.2 illustrates the difference in the workflow of deep learning and other regular machine learning algorithms.

2.2.1 Processamento de imagens e deep learning

Machine learning methods have, for a long time, been used as tools in image processing and scene understanding. The base concept is the same mentioned in the previous section: The learning algorithms use as inputs features extracted from images and learn to interpret them to perform a certain task (such as classification, object detection, etc.).

In classical machine learning, the biggest challenge with processing images is the modeling of the features. Since the feature extraction process is done manually, and images are a more complex kind of input than *e.g.* structured tabular data, a lot of different techniques have to be used together to perform simple tasks. In the classification system presented in Fig. 2.2, different algorithms would have to be implemented for each input to determine, in this case, the color and shape (which are the modeled features) of

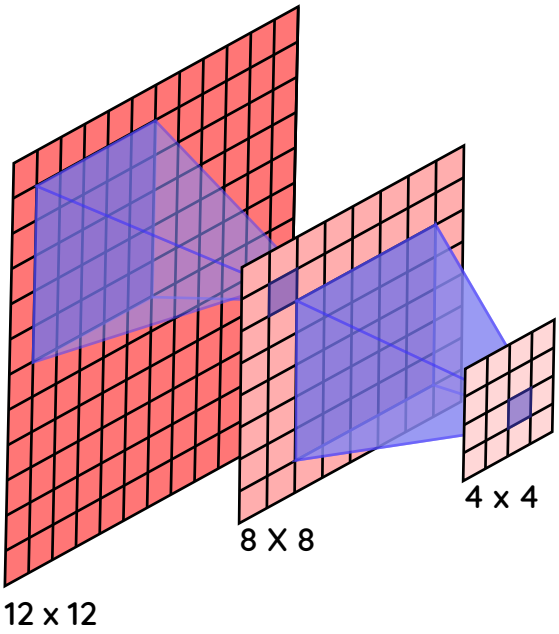


Figure 2.3: Illustration of the convolution operation. The kernel filter (blue) of size (5×5) sweeps the input (red) extracting features, producing a smaller feature map that serves as the input for subsequent layers.

the subject in each image.

The use of deep learning in image-related tasks provides the opportunity to solve these problems at a higher abstraction level. By using deep learning, it is possible to model problems with regards to the end goal: classification, detection, or segmentation. By providing good quality examples to the network during training, it is no longer necessary to manually address specific features from the input. This advantage comes with the pitfall of no longer being able to completely understand the decision process of the prediction, given that most features learned by a neural network are not interpretable easily. The significant improvement in accuracy and ease of implementation, however, tends to outweigh this issue.

The success of deep learning in image processing and computer vision is, in larger part, a product of the use of convolutional layers. The convolution operation performed in 2D images can be summed up to a kernel filter, which is a 2D array (necessarily smaller than the input) that sweeps the input performing multiplication operations with the corresponding input values. The results for each position of the kernel are summed up and result in a single value. Due to that, naturally, the output of each convolutional operation is spatially smaller than the input. This process is interpreted in Fig. 2.3, which shows the reduction of spatial dimensions across three convolutional layers.

The goal of convolutional layers is to "compress" information and extract relevant features. A convolutional neural network usually is made up of multiple sequential convolutional layers. Each layer performs the convolution operation over the output of the

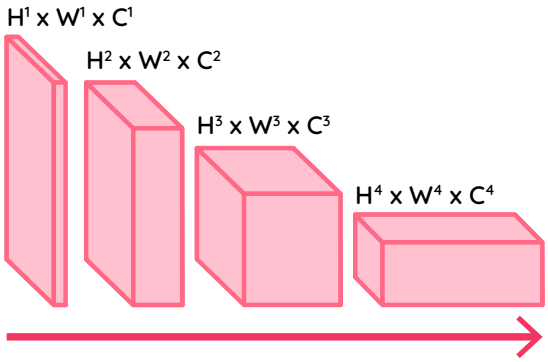


Figure 2.4: Four sequential convolutional layers. Spatial height and width are represented by H and W respectively. The channel dimension is represented by C and illustrated as the depth of each layer. An usual CNN architecture will follow the rules: $H^{i+1} < H^i$, $W^{i+1} < W^i$, and $C^{i+1} > H^i$ for the i_{th} layer of the network.

previous layer. This process can make the layers learn progressively higher-level features. For example, the first convolutional layer in a network can specialize in locating the geometrical edges of an object, which are low-level features. A second or third layer could specialize in learning how to relate these low-level shapes with its surroundings, performing more high-level abstractions. This process repeats itself to an extent where it is no longer possible for human analysts to interpret the learned relationships.

In the previous example, imagine that a filter learns to recognize round shapes. If a square shape is also of interest to the task we are performing, and we only have one filter, we can never learn to recognize squares without modifying the filter and forgetting how to detect round shapes. Hence each convolutional layer usually employs multiple kernels, with each one learning different weights and specializing in distinct features. Traditionally, as the network increases in its depth, we also increase the number of filters in each layer to capture more and more high-level features. The number of filters in a layer is referred to as the layers "channels", and this methodology describes the base for modern CNN architecture. The relationship between spatial size and number of channels is depicted in Fig. 2.4.

In CNNs, in addition to convolutional layers there are two other core elements: **Pooling layers** and **activation functions**. Activation functions are used to introduce non-linearity to the network, allowing it to learn more complex relationships between features. In neural networks, it is considered the norm to apply an activation function to every linear transformation (such as convolutions). Pooling layers serve the function of reducing the spatial size of the feature maps. Unlike convolutions, which also reduce the size but are simultaneously extracting features, pooling layers only downsample the data with the primary goal of reducing the computational cost of a network. Pooling layers can be one of two kinds: Max-pooling or average-pooling. They work as their names suggest, with the max-pooling layer sampling data by taking the maximum value in a range of the input maps, and average-pooling doing the same by taking the average of the values. Pooling layers can improve the representation power of relevant features while reducing

the amount of more noisy data.

With what we discussed so far on the number of convolutional layers and their respective numbers of filters, it is easy to see how a CNN can quickly accumulate a very high number of parameters. For example, the well-established CNN architectures AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), Inception (SZEGEDY *et al.*, 2015), and ResNet-152 (HE *et al.*, 2016) have respectively 62, 138, and 60 million parameters in total. Therefore training CNNs with small data sets can easily cause overfitting.

To have a trained network that performs with adequate accuracy, and possesses enough generalization power to be deployed in the real world, it is necessary to have high volumes of good quality and representative data. This is why the field has seen an increasing number of large-scale image data sets in various niches. One of those is the gaze estimation domain. In Section 3.1.2, we describe in details the most relevant and diverse data sets for this field, including those we eventually used to train and evaluate our proposal.

2.3 MECANISMOS DE ATENÇÃO

Our proposal combines self-attention with regular convolutional layers to extract spatially-aware features from images. Bare this in mind, we introduce here the basic concepts related to attention mechanisms in neural networks. The goal of this section is to briefly go over the history, and explain key theoretical and conceptual aspects of attention. We do so by exploring published papers that eventually led to the creation of the attention-augmented convolutions (BELLO *et al.*, 2019) used in our experiments. The scope of this brief study is defined by the motivation behind the creation of attention in the first place. We cover the basics of recurrent neural networks for sequence modeling, attention mechanisms to aid recurrent networks, self-attention to improve sequence embeddings, the Transformer network, and finally, attention-augmented convolutions.

2.3.1 Redes neurais recorrentes para modelagem de sequências

In regular neural networks, the degree to which is possible to relate elements from an input sequence is usually spatially constrained. That is to say, the larger the input space, the harder it is to model dependencies in an informed manner. For example, in machine translation, the input is a sequence of words and the output consists of several predictions (one for each corresponding word). A common way to address this task is to use an encoder-decoder architecture, where the encoder creates an intermediate representation of the sentence, and the decoder uses that representation to build an output sentence in the target language. It follows that the larger the sentence, the harder it will be to deliver an accurate translation given that the encoder is over-burdened with encoding the entire source sequence into a fixed-length vector (BAHDANAU; CHO; BENGIO, 2014).

To ease the burden of representing long sequences in encoder-decoder architectures, they can be combined with Recurrent neural networks (RNNs). RNNs are a class of neural networks particularly well-suited for sequential tasks. At a given timestep t, the output $\hat{y}(t)$ of a single neuron will be given by the input $x(t)$ and also by the hidden state

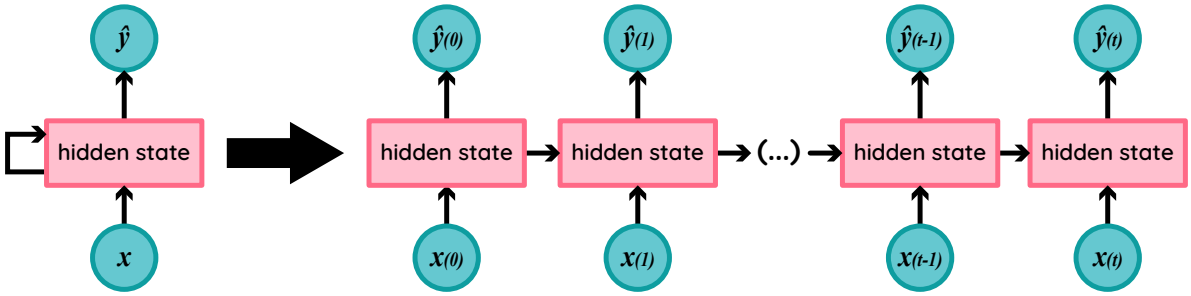


Figure 2.5: Typical structure of an RNN illustrated by a the forward pass on a single neuron. On the left, the compressed version with the recurrence indicated on the hidden state. On the right, the expanded version of the network where it is possible to see the hidden states of past timesteps influencing the output of future timesteps.

$h(t-1)$ (LIPTON; BERKOWITZ; ELKAN, 2015). In essence, for a sequence-like input, past input data will have a direct impact on the inference of future output data. This is illustrated in Fig. 2.5.

Modern examples of RNNs being used to augment encoder-decoder frameworks are the *Long short-term memory* (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) and *Gated Recurrent Unit* (GRU) (CHUNG *et al.*, 2014) units. Given an input vector $x = (x_1, x_2, ..., x_T)$, and a series of RNN states defined by $h_t = f(x_t, h_{t-1})$, in a typical application the encoder would build the intermediate context representation $c = q(h_1, h_2, ..., h_T)$ and each word of the output sequence would be chosen by a positional probability modeled as:

$$p(y_t \mid \{y_1, ..., y_{t-1}\}, c) = g(y_{t-1}, s_t, c)\,,^1 \tag{2.3}$$

where g is a non-linear function (neural network) and s_t is the RNN's hidden state. Objectively, each predicted word is a function of the overarching context, the current state of the recurrent unit, and the previous prediction.

2.3.2 Mecanismos de atenção

While the simplistic RNN approach presented before retained state-of-the-art status for a long time, the information bottleneck of the encoder persisted. The implementation of attention mechanisms in (BAHDANAU; CHO; BENGIO, 2014) presented a solution to this problem by using a bi-directional RNN: In the encoder, each element in the input sequence x is assigned an annotation vector h (made up of the concatenated forward and backward hidden states from the bi-directional RNN). This annotation vector contains information about the entire sequence but strongly focuses on the context immediately around that element.

In contrast with the naive recurrent model where a single context vector c was used to provide overarching information about the input, the context vector c_i is now calculated

[1] Mathematical formulations adapted from (BAHDANAU; CHO; BENGIO, 2014).

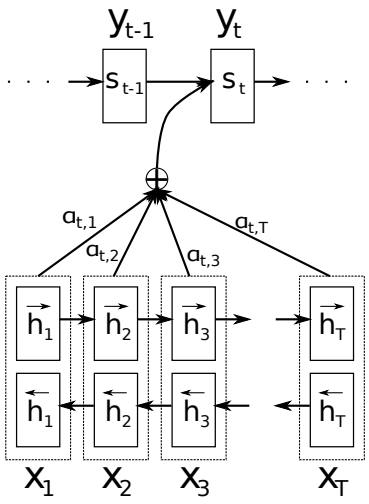


Figure 2.6: Attention mechanisms are incorporated within a bi-directional RNN in a translation model. The illustration shows the prediction step of the t_th word (y_t) from an input sentence ($X_1, ...X_T$) where the context is inferred from attention components ($a_1, ...a_T$). Image taken from (BAHDANAU; CHO; BENGIO, 2014).

relative to the element's position by leveraging the annotations. The new formulation is:

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j \,, \tag{2.4}$$

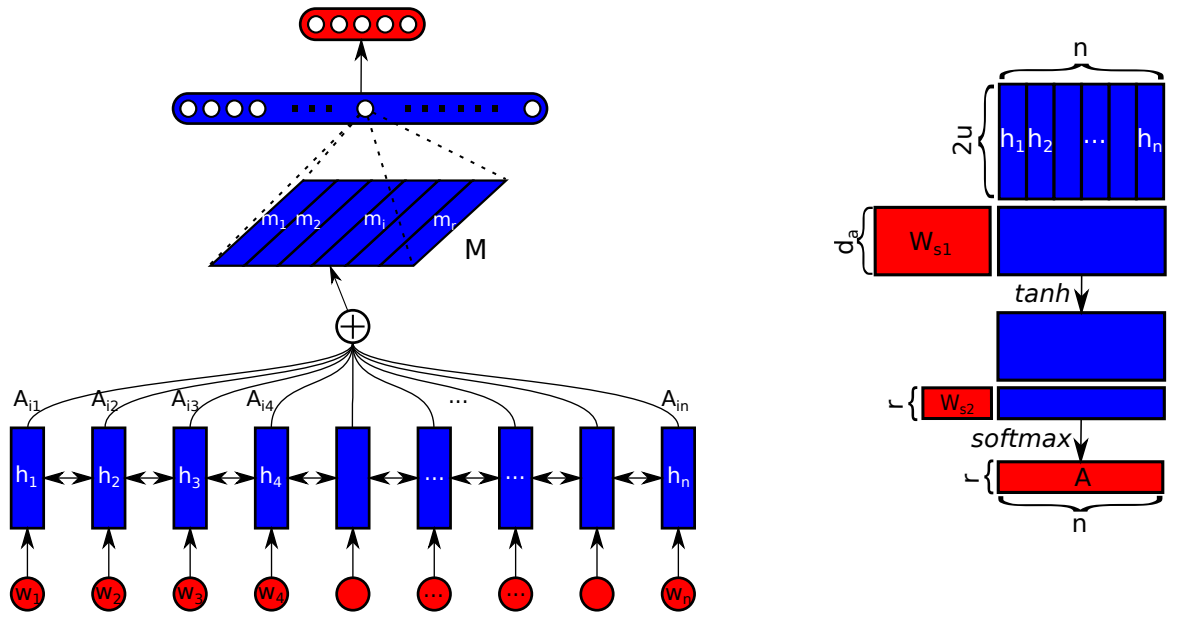
where a_{ij} is the weight assigned to each annotation, calculated by scoring how well the input in position j and the output at position i match (details about the scoring function fall outside the scope of this document and can be found in (BAHDANAU; CHO; BENGIO, 2014)).

The use of dynamic contexts means that information is no longer centralized in a single fixed-length vector, and the decoder can selectively access relevant information from the annotations according to the input element's position. A graphical representation of the prediction step is illustrated in Fig. 2.6.

2.3.3 Auto-atenção

The attention mechanism presented by (BAHDANAU; CHO; BENGIO, 2014) rapidly established itself as a staple in NLP tasks due to its clear advantages over traditional methods in sequence modeling. The concept of self-attention introduced in (LIN *et al.*, 2017) is proposed as an improvement over the original attention mechanism, and it stands out as one of the most relevant to this dissertation.

Proposed as a way to improve sentence embeddings for tasks like sentiment analysis, the self-attention mechanism modifies the previously explained attention mechanism in two main aspects. First, the embedding vector is replaced with a 2D matrix, with each row attending on a different part of the sentence. This change increases the positional



(a) (b)

Figure 2.7: Illustration of the sentence embedding process of a a sample self-attention model. (a) shows the inputs being parsed into representation matrix M via the weighted sums of hidden states from a bidirectional recurrent unit $(h_1, ..., h_n)$. (b) illustrates the computation of attention weights $(A_1, ..., A_n)$. Images taken from (LIN *et al.*, 2017).

awareness of the representation matrix considerably in comparison with 1D embedding vectors (it is easy to see here already how this has the potential to be expanded into spatial awareness for 2D data, i.e. images).

Second, the concept of self-attention is introduced. Self-attention is a modification of traditional attention mechanisms capable of extracting different aspects of the input sequence into various vector embeddings. Self-attention accomplishes this by reformulating the previously established attention weights a and the LSTM hidden-states h into matrices so that the resulting sentence embedding is now $M = AH$ where M is a 2-D matrix. These modifications can, according to the authors, help the model "better disentangle the latent information from the input sentence" (LIN *et al.*, 2017). An example of a self-attention recurrent model applied to the translation task is illustrated in Fig. 2.7.

2.3.4

The Transformer model proposed in (VASWANI *et al.*, 2017), like the previously discussed methods, was initially designed for NLP (particularly machine translation) tasks. Although it follows the general encoder-decoder structure, the Transformer did away with recurrence completely, relying instead entirely on self-attention to compute input embeddings and outputs.

In (VASWANI *et al.*, 2017), attention is described as "mapping a query and a set of

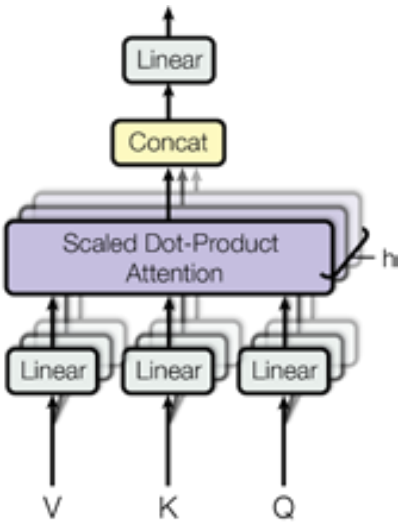


Figure 2.8: Multi-headed attention performed on a set of inputs. $Nh = h$ attention heads compute the output in parallel and are concatenated before passing to the output layer (in this case, a fully connected linear unit). The formulation for "Scaled Dot-Product" attention computation performed by the Transformer falls out of the scope of this document but follows the same principle of the self-attention layers explained in this chapter. Image taken from (VASWANI *et al.*, 2017).

key-value pairs to an output, where the query (Q), keys (K), values (V), and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key". This is essentially the same concept covered previously in Section 2.3.2, but without the inherent dependence on a recurrent network's hidden states to compute the weighted sum.

Although there are many concepts introduced by the Transformer network, one that is highly relevant to this work is multi-headed attention. Instead of computing the attention output in a single pass, the multi-headed attention process is to linearly project the attention inputs (Q, K, V) Nh times using different learned linear projections to the previously established d_q, d_k, and d_v dimensions respectively. In each projection, the attention step is performed in parallel. The outputs are concatenated and back-projected to the output dimensions.

Figure 2.8 illustrates the multi-headed attention concept. Multi-headed attention helps the model to simultaneously attend to data from different embedding sub-spaces in different positions. This is a desirable effect, which is inhibited by the averaging of outputs performed in regular attention operations as claimed by (VASWANI *et al.*, 2017).

2.3.5 Attention-augmented convolutional layers

The attention-augmented convolutional (AAConv) layers proposed in (BELLO *et al.*, 2019) are one of the base building blocks of the work described in this document. These layers adapt many of the concepts previously discussed in this section and apply them to images.

Considering an input image I with height, width and channels defined as (H, W, C), respectively, the AAConv operation is formulated as:

$$AAConv(I) = Concat\left[Conv(I), MHA(I)\right] , \tag{2.5}$$

where the matrix $I \in \mathbb{R}^{HW \times C}$ is the flattened input image, $Conv$ is a regular convolution operation and MHA is the multi-headed attention component.

The output of the MHA component is the concatenation of the outputs from each of the attention heads, which work similarly to the multi-headed component of the Transformer architecture presented in Fig. 2.8.

2.4 CLOSURE

This chapter presented some of the context necessary to the understanding of our proposal. Next chapter, these concepts are seen in practice when we discuss relevant related works published in the areas of appearance-based gaze estimation and attention mechanisms.

CHAPTER 3

RELATED WORK

Contents

Gaze estimation has been an active area of research for a long time, with early works relying on detailed theoretical modeling and complicated physical setups (ZHU; JI; BENNETT, 2006; OHNO; MUKAWA, 2004; NOUREDDIN; LAWRENCE; MAN, 2005; TALMI; LIU, 1999) to obtain usable results. Appearance-based gaze estimation is an attractive alternative given its relative implementation simplicity when compared to model-based approaches. The use of classical machine learning techniques (MARTINEZ; CARBONE; PISSALOUX, 2012; WILLIAMS; BLAKE; CIPOLLA, 2006; TAN; KRIEGMAN; AHUJA, 2002) as mapping functions to regress gaze angle vectors from input images showed promise, but its practical application potential was still limited due to its performance in unconstrained conditions.

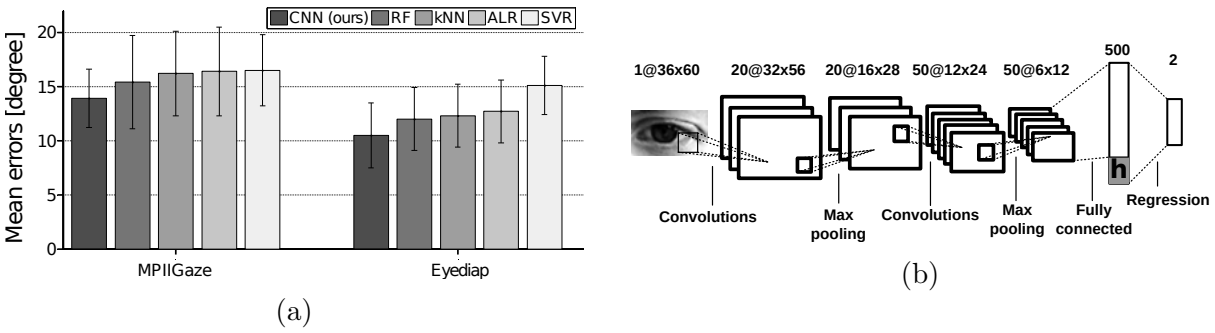
Similar to many image-related tasks, gaze estimation greatly benefited and started gaining traction with the evolution of deep learning in computer vision over the last decade. The rapidly increasing accuracy of methods that use deep learning and the ubiquity of low-cost cameras have since then further reinforced appearance-based as the most promising path towards unconstrained gaze estimation in-the-wild.

In this chapter, we go over important published works in the appearance-based gaze estimation domain, focusing on those that use convolutional neural networks as part of the gaze estimation pipeline. Additionally, we address the use of attention mechanisms in computer vision as a recent trend in deep learning research. Finally, we present a brief comparison of relevant previously published methods with the one proposed in this document.

3.1 GAZE ESTIMATION

3.1.1 Appearance-based gaze estimation

Zhang *et al.* (2015) were the first to apply CNNs to the task of appearance-based gaze estimation. Good quality training data is one of the biggest concerns when using CNNs,



(a) (b)

Figure 3.1: Comparative evaluation results against baseline traditional machine learning methods (a) and architectural details of the CNN used (b). Images taken from (ZHANG *et al.*, 2015).

and most of the available data sets at the time were recorded in heavily controlled laboratory environments. The MPIIGaze data set, released as part of that work, was one of the first large-scale gaze estimation data sets with a focus on unconstrained data captured in daily-life settings. Additionally, experiments presented by the paper compared the mean angular error of the proposed CNN method against traditional machine learning algorithms currently held as state-of-the-art and found that CNNs had the best results on two different data sets (Fig. 3.1a).

Several subsequent publications have built upon the notion of using CNNs to improve accuracy by reducing the mean angular error. The main challenge in proposing a new appearance-based gaze estimation framework is to incorporate domain knowledge into the work so that the model can implicitly or explicitly take into account specific aspects particular to the task to improve its performance and decrease the evaluation error. For example, the subject's head pose is one feature that ideally should be part of any gaze estimation pipeline, given that it directly impacts the gaze direction. The previously mentioned MPIIGaze architecture naively incorporates head pose by concatenating the head angles as numerical values with the extracted features from the eye images (noted as $\boldsymbol{h}$ in Fig. 3.1b). The biggest issue with this approach is that when performing gaze estimation in-the-wild, one would need to have a stage in the processing pipeline to estimate the head pose angles, introducing unnecessary complexity and computational overhead to the system.

The iTracker architecture (KRAFKA *et al.*, 2016) proposed a simpler approach to resolve the head pose issue by using the entire face image as an input to the network. As illustrated in Fig. 3.2, the iTracker uses a multi-input architecture where the eyes and the face are passed through convolutional networks and a binary face grid is used to encode the head's position and scale. The authors state that the face image should be enough for the network to infer the influence of head pose with regards to the gaze direction while the binary grid helps with the gaze projection (since the system's outputs, in this case, are coordinates of a pixel in a mobile phone's screen, and not a direction vector). While the proposal showed promising results, it is worth noting that the use case in this scenario (mobile phones) is a fairly stable one with regards to head pose, and the system

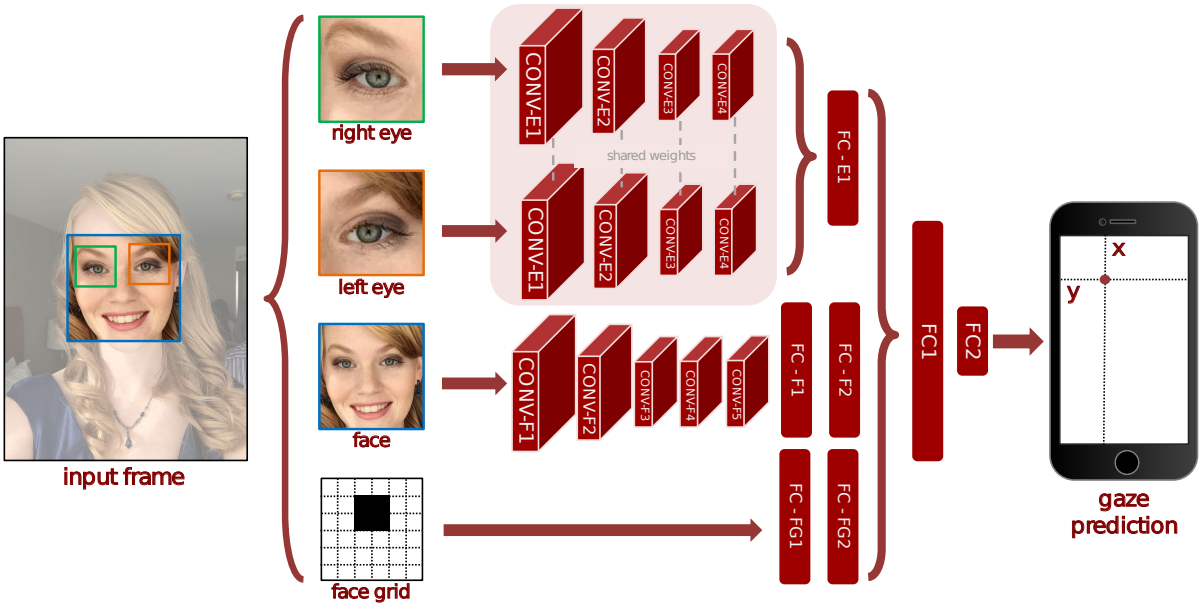


Figure 3.2: Network architecture for the multi-input iTracker gaze estimation model. The right- and left-eye inputs are passed through CNNs with shared weights, the face input has its own separate CNN backbone, and the binary face grid is passed only through a FC layer. The results are concatenated and passed through another FC layer to output the final pixel coordinates. Image taken from (KRAFKA *et al.*, 2016).

does not generalize as well to more complex situations.

The framework proposed in (PALMERO *et al.*, 2018), like the iTracker, does not use explicit numerical values for head pose angles. They also use full-face images as one of the inputs, and the head pose information is further reinforced by the inclusion of facial landmarks coordinates as a separate input (illustrated in Fig. 3.3). While this still introduces a level of complexity to the processing pipeline, it should be more robust for deployment than the head pose angles used in (ZHANG *et al.*, 2015), since facial landmarks are simpler to obtain and sometimes already provided by face detectors. Additionally, the authors propose the use of recurrent CNNs to improve prediction accuracy significantly on continuous inference, remarking that gaze is an inherently temporal phenomenon. The intuition behind this is trivial. The direction where someone is looking at in a particular moment in time directly depends on where they were looking at in a previous moment.

Still on the topic of leveraging head pose information, (ZHANG *et al.*, 2017) proposed an efficient way to extract the implicit data contained in full-face images without the aid of additional input modalities. They propose a spatial-weights mechanism that can be inserted in the CNN backbone and trained in an end-to-end manner with the rest of the network. After training, this mechanism outputs a spatial importance map indicating regions in the input image that is the most relevant to the gaze estimation task.

As shown in Fig. 3.4, the spatial weights map is applied to the input features using element-wise multiplication. The map acts as a guide for the following layers of the network, helping them focus on the most relevant features extracted from the face. Ad-

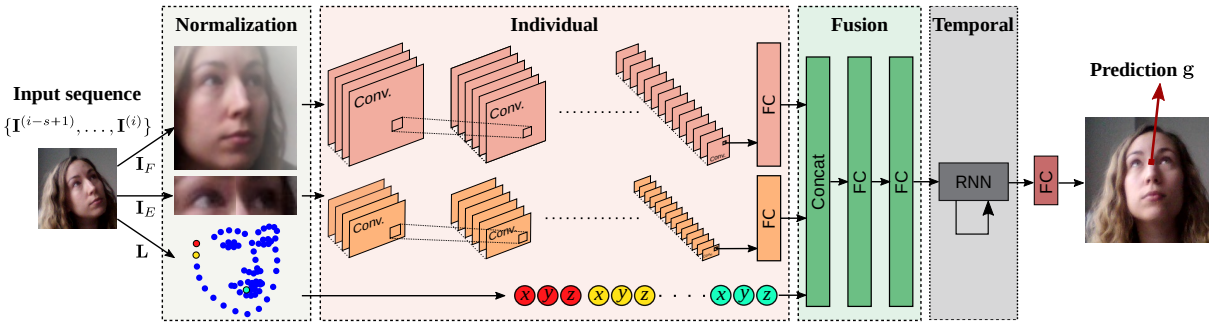


Figure 3.3: Architectural details of the recurrent gaze estimation network. Features extracted from the normalized face and eye images are fused with the coordinates of the facial landmarks through fully connected layers, and fed to a recurrent unit at each time-step. Image adapted from (PALMERO *et al.*, 2018).

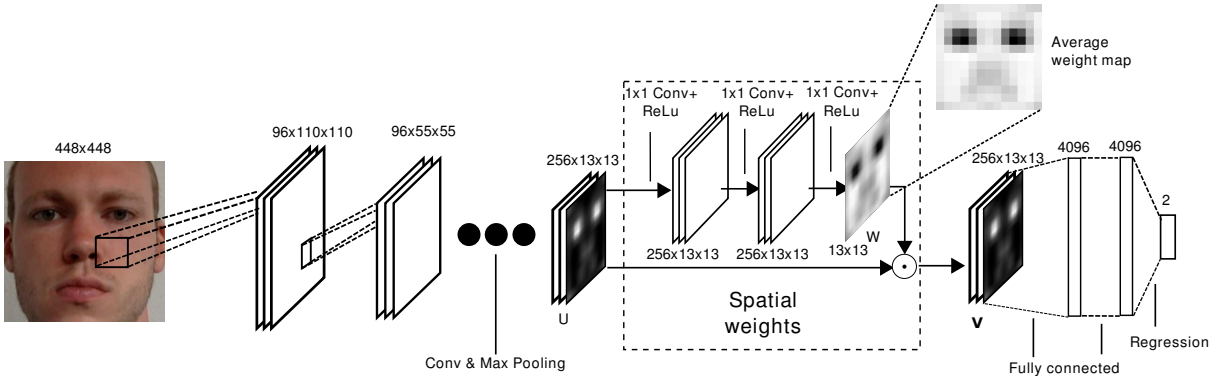


Figure 3.4: CNN with spatial weights mechanism. The feature map U produced from the convolution layers is copied and forwarded through a sequence of 1x1 convolutional layers to produce the spatial weights map W. Both maps are joind via element-wise multiplication to produce the feature map V. Image taken from (ZHANG *et al.*, 2017).

ditional experiments presented by the paper show that this approach was particularly beneficial on images with extreme angles of head pose. The use of facial images as input became more or less the norm in appearance-based gaze estimation, with the most modern works usually using them along with isolated eye patches. One such example is the RT-GENE architecture (see Fig. 3.5), used as a proof-of-concept for the data set proposed in the same paper.

Recent proposals have also been focusing on modeling other domain-specific characteristics of the gaze estimation task to improve accuracy. Cheng *et al.* (2020) remark that often the left- and right-eye images are asymmetric in quality. This phenomenon can happen when one of the eyes is occluded due to extreme head pose or due to poor lighting on one side of the face. To overcome that, Cheng *et al.* (2020) propose a sub-network capable of predicting a reliability for each eye. The output of this sub-network (E-Net in Fig. 3.6) is used by the main network (FAR-Net in Fig. 3.6) to calibrate the results in accordance to the reliability of each eye, relying more on the image with the

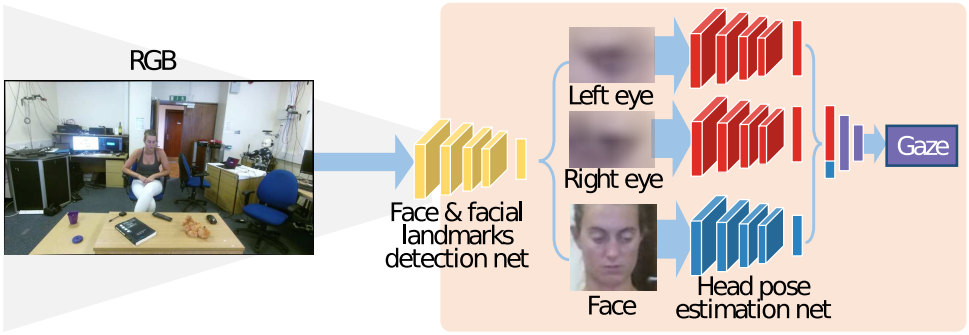


Figure 3.5: Illustration of the network used for inference in the (FISCHER; CHANG; DEMIRIS, 2018) paper. Image adapted from (FISCHER; CHANG; DEMIRIS, 2018).

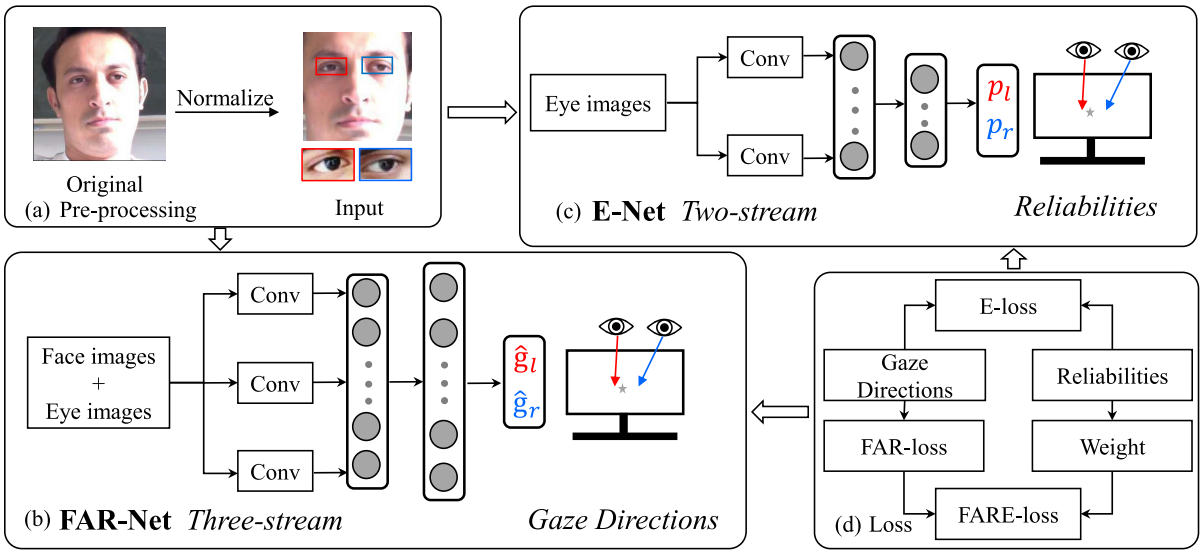
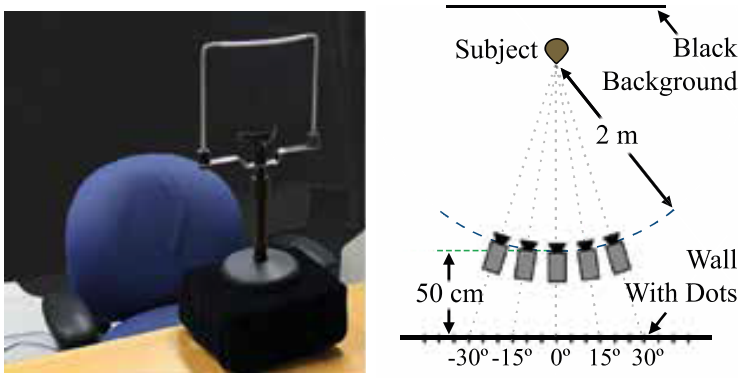


Figure 3.6: Processing pipeline of the FARE-Net components. The normalized inputs are fed to E-Net to compute reliability scores for each eye, and to FAR-Net to compute gaze direction angles. Image taken from (CHENG *et al.*, 2020).

highest quality.

Another example of domain-specific modeling for deep learning is proposed in (CHEN; SHI, 2020). Chen & Shi (2020) note that the downsampling stages of neural networks can easily cause the loss of relevant information. In eye images, slight differences in eye movement can be relevant for gaze direction but are also easily lost between layers due to the reduction of spatial resolution that occurs in CNNs. To tackle this problem, Chen & Shi (2020) propose the use of dilated convolutions as a replacement for some convolutional and max-pooling layers as a way to increase the receptive field of the layers while keeping the number of parameters manageable.



(a) (b)

Figure 3.7: Data collection procedure for the Columbia data set. Physical recording setup (a) and top-down scheme illustration (b). Images taken from (SMITH *et al.*, 2013).

3.1.2 Data sets

Most of the early gaze estimation data sets are composed of high-resolution images with the subjects being close to the camera and in constrained mobility conditions. These settings can lead to unfair biases in the evaluation procedure where a model could show good accuracy in controlled environments but perform poorly in real-world settings. To properly assess the validity of proposed techniques, the training and testing data should approximate as closely as possible to real-world conditions. To this effect, newer data sets have considered many different aspects during their data collection process, such as head pose variation, greater distances between subject and camera, lower quality images, etc.

Specifically designed for gaze tracking and eye-contact detection, the Columbia data set (SMITH *et al.*, 2013) was the first to be released that could potentially be used as a relevant benchmark for appearance-based gaze detection. It contains 5880 images collected from 56 subjects. The images were collected in sessions where the subjects directed their gaze towards 21 different target locations while moving their heads through 5 pre-determined positions. The subjects have a seat in front of a black background with the head position fixated by a chin rest. Five cameras (each representing a different head pose) would collect images as the participants moved their eyes through a grid of targets in front of them. This is illustrated by Fig. 3.7. As previously mentioned, the collection conditions were very controlled (uniform background, illumination, fixated head, high-resolution images (5184×3456). While this can be enough for the intended eye-contact detection application, it is trivial to see why the data can't accurately represent in-the-wild scenarios when inspecting the images (Fig. 3.8).

The EYEDIAP data set (MORA; MONAY; ODOBEZ, 2014), released shortly after the Columbia data set, is a significantly more challenging data set. It is a collection of 94 videos with 16 different subjects. In a similar fashion to Columbia data set, participants were asked to sit in front of the recording set up and look at targets. The sessions were split into 3 different modalities: **Discrete screen target** — where a target was displayed

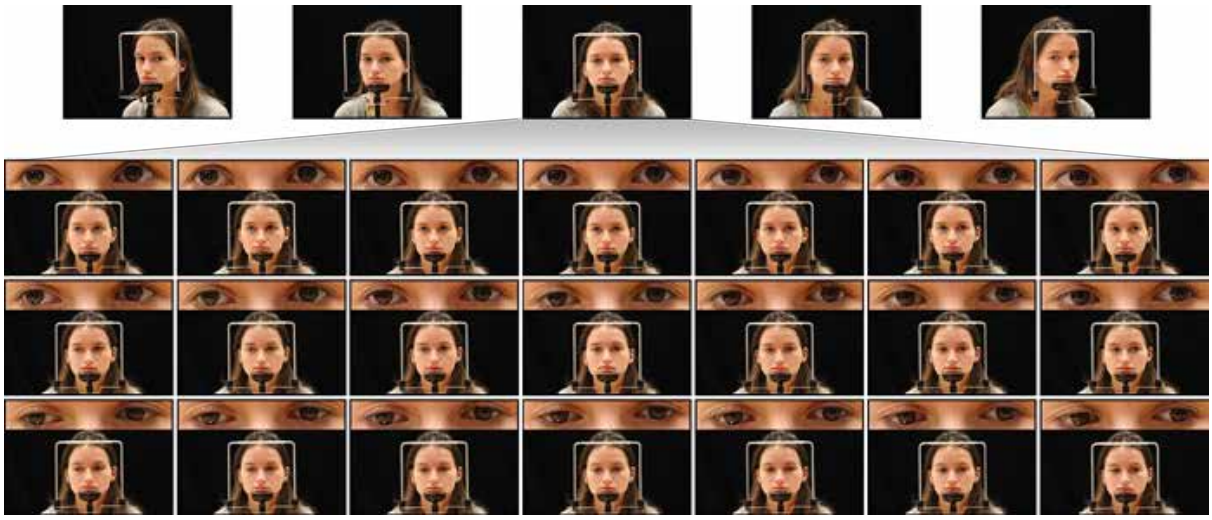


Figure 3.8: Data sample from the Columbia data set. The top row represents the 5 different head poses available on the data set, and the grid below represents the 21 gaze targets available for each head pose. Image taken from (SMITH *et al.*, 2013).

Figure 3.9: Data collection setup for the EYEDIAP data set. Image taken from (MORA; MONAY; ODOBEZ, 2014).

in regular intervals on random locations on a screen, **continuous screen target** — in which the target moved along random trajectories in the screen, and **3D floating target** — where a small ball was moved along the 3D space between the participant and the camera with the assistance of a thin thread. Figure 3.9 shows the data collection setup in action.

EYEDIAP is significantly more adequate for appearance-based gaze estimation than the Columbia data set, given the particular fact that it provides unconstrained head pose. It still falters, however, when it comes to the background and light conditions, which are still uniformly distributed across the data set.

The UT Multiview data set is composed of video segments from 8 different angles recorded from 50 different participants using again a fixed chin-rest to standardize head pose. A grid of 160 cells placed 60 cm away from the subjects was used to randomly

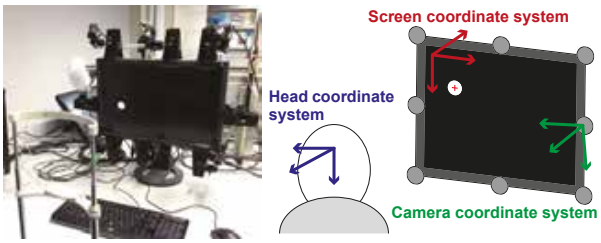


Figure 3.10: Data collection setup for the UT Multiview data set. Image taken from (SUGANO; MATSUSHITA; SATO, 2014).

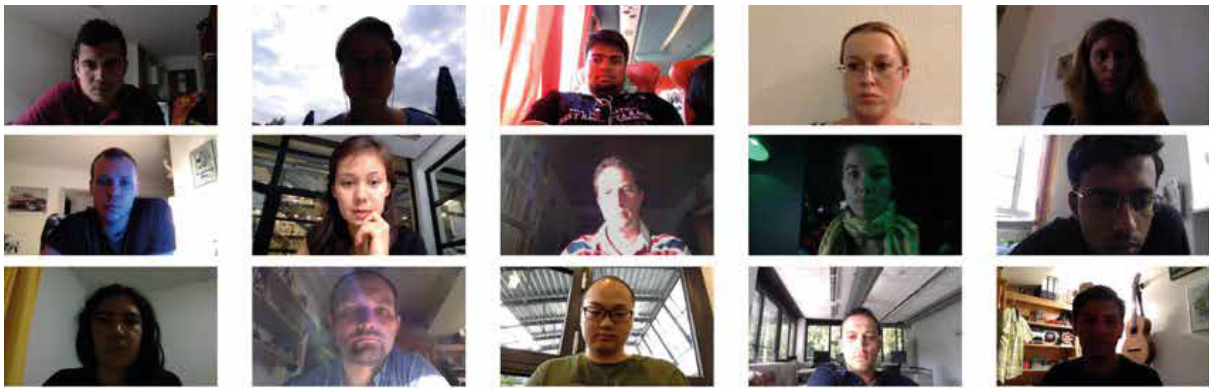


Figure 3.11: Samples from the MPIIGaze data set showcasing the large variation in environmental settings present in the data. Image taken from (ZHANG *et al.*, 2015).

display targets (Fig. 3.10. Again, the data collected is heavily reliant on a controlled environment, not being the most suitable for meaningful evaluations of appearance-based gaze estimation methods.

MPIIGaze (ZHANG *et al.*, 2015) was the first to provide unconstrained data for gaze estimation in-the-wild. The data set was collected from various sessions featuring 15 subjects (9 males and 6 females, with 5 of them wearing glasses). In contrast with previous data sets recorded in strict lab conditions, the MPIIGaze images were captured during day-to-day use of the subjects' laptops (targets occasionally displayed at random positions on the screen). The recorded data contains a large number of different conditions (Fig. 3.11) with regards to recording locale (inside and outside), illumination, and overall recording quality, making it one of the best data sets to validate gaze estimation methods for use in real-world conditions.

Some of the more recent data sets focus on a particular niche of applications, such as the RT-GENE (FISCHER; CHANG; DEMIRIS, 2018), TabletGaze (HUANG; VEERARAGHAVAN; SABHARWAL, 2017), and GazeCapture (KRAFKA *et al.*, 2016) data sets. RT-GENE aims at representing large distances between subjects and cameras/targets. The larger the distance, the more relevant is the effect of head pose in gaze estimation, which was a problem not directly addressed by the data sets available. TabletGaze and GazeCapture focus on the opposite problem, close subject-camera distances on mobile devices (specifically handheld tablets on the former). These data sets are not as

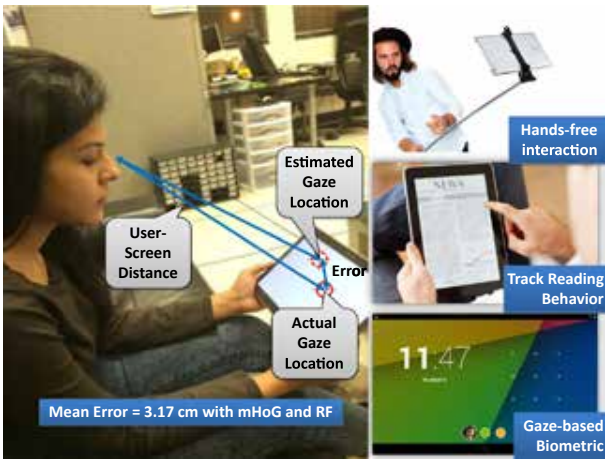


Figure 3.12: Functional illustration of the nature of the TabletGaze data set. Image taken from (HUANG; VEERARAGHAVAN; SABHARWAL, 2017).

relevant to our problem given their limited scope, which makes them more suitable for 2D gaze estimation tasks as illustrated in Fig. 3.12.

3.2 ATTENTION MECHANISMS FOR COMPUTER VISION

Section 2.3 presented an overview of attention mechanisms in deep learning, their motivations, and inner workings. In CNNs for image-based tasks, attention mechanisms can help encode long-range dependencies and context in a way regular convolutions are not capable of doing. In this section, we go over applications of attention mechanisms in computer vision specifically, briefly touching on how they relate to our motivation for this work.

Many works have proposed a modular approach to integrating attention to convolutional neural networks commonly used in image-related tasks. The motivation for this is that a stand-alone component capable of being effortlessly inserted into existing architectures to improve their performance has good potential for practical applications with minimal modification effort. The Squeeze-and-Excitation (SE) blocks proposed in (HU; SHEN; SUN, 2018) are one such example. SE blocks are a gating mechanism that attempts to improve the visual representation capabilities of the convolutional layers by explicitly leveraging channel-wise relationships in feature maps. For any given convolution or set of convolutions, an SE block can be used to re-calibrate features by learning global contexts from the channels and using that information to emphasize relevant information.

Figure 3.13 illustrates how an SE block behaves when inserted into a CNN. First, features outputted from a convolutional layer are *squeezed* across their spatial dimensions to produce a purely channel-wise descriptor. Then, an *excitation* operation, learned for each channel on global contexts, acts as the attention gating mechanism by selectively exciting or suppressing the information. The original feature maps are then re-weighted accordingly and become the output of the SE block. This is a simple concept that can

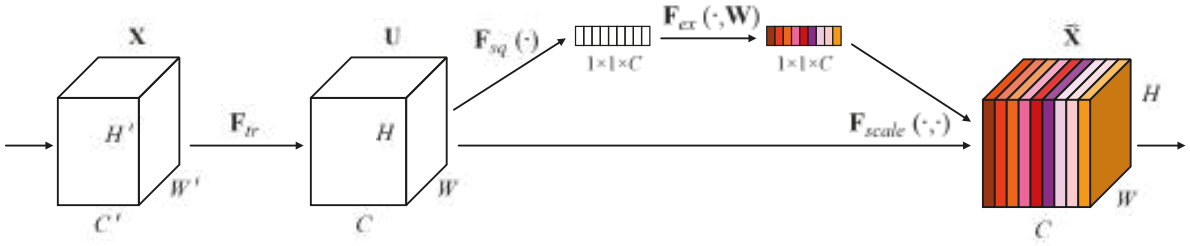


Figure 3.13: SE block used on the output (**U**) of a convolutional layer ($\mathbf{F_{tr}}$). The squeeze operation ($\mathbf{F_{sq}}$) reshapes the features to the channel space, and the excitation operation ($\mathbf{F_{ex}}$) assigns weights to each channel based on learned relationships. Image taken from (HU; SHEN; SUN, 2018).

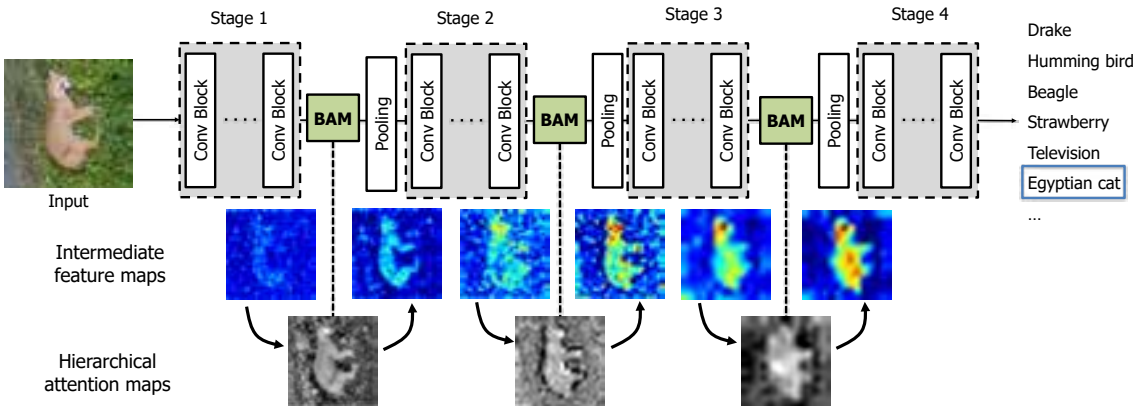


Figure 3.14: Illustration of BAM placement in a feed-forward convolutional neural network. Each Bottleneck Attention Module hierarchically improves the input features of eack stage of the network. Image taken from (PARK *et al.*, 2018).

potentially extend any network's ability to encode channel-wise relationships.

The Bottleneck Attention Module (BAM) proposed in (PARK *et al.*, 2018) is another example of an attention-based drop-in module for convolutional neural networks. The concept behind BAM is not that different from the SE blocks mentioned previously, but unlike SE blocks, BAM exploits attention on both spatial and channel-wise dimensions. As illustrated in Fig. 3.14, BAM is placed in every bottleneck stage of a CNN. At each stage, the module is capable of constructing hierarchical attention by separately computing spatial and channel attention maps, then joining them and performing element-wise multiplication on the input feature map. The use of both spatial and channel cues leads BAM to outperform SE blocks. Placing the modules exclusively on the bottleneck stages also creates less overhead to the overall computational cost of the network.

The Convolutional Block Attention Module (CBAM) (WOO *et al.*, 2018) is a very similar proposal to BAM. Both compute channel and spatial attention separately from an input feature map and use the combination of these to refine the output features. The main differences are that while BAM performs these operations in parallel over the

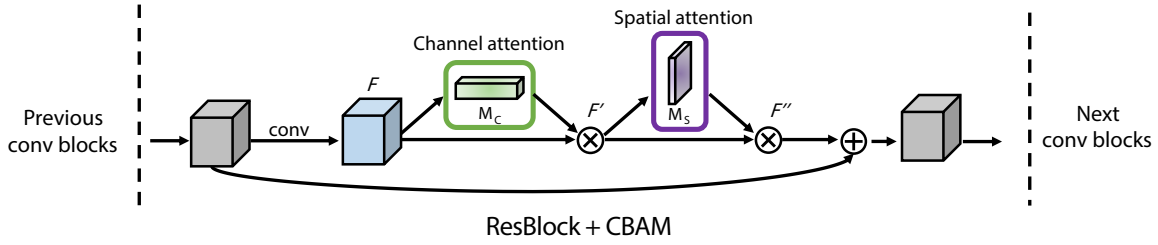


Figure 3.15: CBAM integrated into a ResNet (HE *et al.*, 2016) block, showing the sequential application of channel-wise (M_C) attention and spatial (M_S) attention to a feature map (F). Image taken from (WOO *et al.*, 2018).

same feature map, CBAM performs them sequentially (as depicted in Fig. 3.15) using the combination of the raw input features and the channel attention module as the input to the spatial attention module. Additionally, while BAM is applied in every bottleneck stage of a CNN architecture, CBAM is applied to every convolutional block.

The SE, BAM, and CBAM units mentioned so far are all different approaches with the same principle: a gating mechanism based on attention capable of being added to existing CNN architectures, allowing them to better exploit context and emphasize relevant features. The attention-augmented convolution (AAConv) unit proposed in (BELLO *et al.*, 2019), however, is not meant to be inserted into existing CNNs, but rather replace convolutional layers entirely.

In regular convolution layers, inter-pixel correlation is usually spatially constrained by the convolutional kernel. This restriction limits the degree to which it is possible to relate distant sections from an image that could have relevant relationships. In (BELLO *et al.*, 2019), the principle of multi-headed self-attention from the Transformer network presented in Section 2.3.4 is adapted for 2D inputs, resulting in a hybrid layer with attention and convolution operations performed in parallel.

Similar to what is done in Transformer networks with 1D sequences, attention-augmented convolutions use self-attention to handle pixel matrices. Each pass through an AAConv layer can be split into two main parts: The first one through a regular convolutional layer, while the second through a multi-headed attention layer. The outputs of each individual attention-head are concatenated and projected onto the original spatial dimensions of height and width of the input. In the end, the results from both passes of the convolutional and the multi-headed attention layers are concatenated, forming spatially-aware convolutional feature maps from the input image. An overview of this whole process is illustrated in Fig. 3.16.

By expanding the neural network principle of long-distant spatial relationships, it is possible to achieve positive effects in straight-forward classification tasks across a range of different architectures (BELLO *et al.*, 2019). These layers are compatible with current established deep network architectures, being able to completely replace regular convolutional layers in their architectures. Unlike the BAM and the CBAM, which use attention to refine existing convolutional feature maps, self-attention augmented convolutions cre-

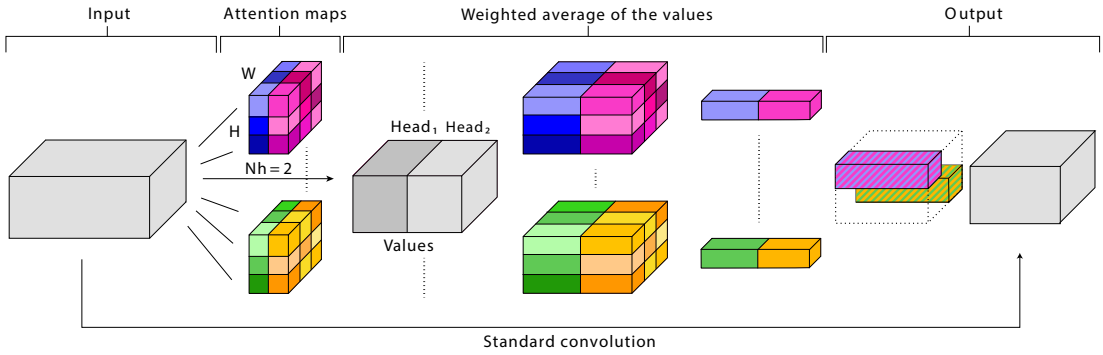


Figure 3.16: Attention augmented convolutional layer. Nh attention maps are computed for every (h, w) location of the input. The self-attention step consists of performing weighted averaging in these maps, then combining the results by concatenating, reshaping, and mixing (1×1 convolution). The resulting feature map is then concatenated with a feature map obtained from a regular convolution performed directly on the input, producing the final output.

Method	3D gaze output	Full-face as input	Eye as input	Multimodal inputs	Spatial awareness	Attention augmented
MPIIGaze (ZHANG *et al.*, 2015)	✓	–	✓	–	–	–
iTracker (KRAFKA *et al.*, 2016)	–	✓	✓	✓	–	–
Spatial Weights (ZHANG *et al.*, 2017)	✓	✓	–	–	✓	–
RT-Gene (FISCHER; CHANG; DEMIRIS, 2018)	✓	✓	✓	–	–	–
Recurrent CNN (PALMERO *et al.*, 2018)	✓	✓	✓	✓	–	–
Dilated Net (CHEN; SHI, 2020)	✓	✓	✓	–	–	–
FAR-Net (CHENG *et al.*, 2020)	✓	✓	✓	–	–	–
ARes-Gaze (Ours)	✓	✓	✓	–	✓	✓

Table 3.1: Summary of the state-of-the-art on appearance-based gaze estimation.

ate new attention maps to be fused with their convolutional counterparts. This process allows the network to learn more spatially-aware representations, potentially presenting accuracy gains.

3.3 CLOSURE AND RELATION WITH OUR WORK

In this chapter, we listed several appearance-based gaze estimation methods that inspired the proposed ARes-Gaze architecture, which is detailed in the next chapter. Table 3.1 presents an itemized list of these methods that we consider similar enough to be compared to ours.

ARes-Gaze stands out especially with regards to the explicit inclusion of spatial awareness to the prediction network. The intuition behind this is similar to that presented in (ZHANG *et al.*, 2017), where the spatial weights mechanism created what could be called a rudimentary approximation of an attention map before the prediction stage of the network. In contrast, our work uses AAConv layers with multiple attention maps in each layer to accomplish the same objective in a more robust and informed manner.

Our overall architecture follows the most recent appearance-based frameworks (KRAFKA *et al.*, 2016; FISCHER; CHANG; DEMIRIS, 2018; PALMERO *et al.*, 2018; CHEN; SHI, 2020; CHENG *et al.*, 2020) in that it uses both eyes and face images as inputs. However, we use a novel input modality for the eye images by stacking crops from the left and right eyes on top of each other before feeding them to the network. This method shows better results on evaluation with equal or lower computational cost than the commonly used alternatives (further discussions and experimentation details on this topic are in Chapter 4 and Chapter 5). Additionally, we do not require multi-modal inputs such as the facial landmark key points used in (PALMERO *et al.*, 2018).

CHAPTER 4

ESTIMATING GAZE WITH ATTENTION-AUGMENTED CONVOLUTIONAL NETWORKS

Contents

The core of our proposal is the use of attention-augmented convolutional layers to improve the accuracy of appearance-based gaze estimation. This chapter covers the structural details of the ARes-14 network, which is an attention-augmented backbone conceived to be used as a feature extractor augmented by attention in every convolutional layer, briefly describing the base architecture and design. Then, we address the ARes-Gaze, our gaze estimation framework based on the ARes-14 architecture.

4.1 ARES-14: AN ATTENTION AUGMENTED CONVOLUTIONAL NETWORK

In appearance-based gaze estimation, shallow CNNs can be sufficient as long as the task is performed in relatively constrained conditions (FISCHER; CHANG; DEMIRIS, 2018). That is to say, it is desirable to limit the range of head pose variation and to have relatively short distances between subject and camera. Although these constraints are intrinsic to some available gaze estimation data sets, it is not reasonable or realistic to expect these conditions to be a guarantee for in-the-wild applications. These conditions can be simulated in more challenging data by pre-processing and applying normalization procedures on the input images (see Section 5.1.2 for more details). The use of these strategies helps maintain the CNN's computational complexity low and allows us to train with more structured data, while still performing well in more complex environments by transforming the input data before sending it through the prediction network during inference time.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 4.1: Detailed layer structure of the 18-, 34-, 50-, 101-, and 152-layer variants of the ResNet architecture. Figure adapted from Table in (HE *et al.*, 2016).

The use of shallower networks is of particular importance given the significant computational overhead of training with self-attention in convolutional networks (see (BELLO *et al.*, 2019) for a more detailed discussion on this topic), which are an integral part of our proposal. In this section, we first describe the ResNet (HE *et al.*, 2016) architecture that was used as a base for our ARes-14 model, then we go over the ARes-14 architecture itself, justifying the design decisions made during its conception.

4.1.1 The ResNet architecture

ResNet (HE *et al.*, 2016) is a widespread and well understood general-purpose CNN, turning it an ideal candidate for a baseline comparison against self-attention augmentation. The general ResNet architecture is composed of one input stem, four convolutional blocks, and an output layer. The ResNet architecture is also flexible in its depth, and the authors of the original paper present versions with 18, 34, 50, 101, and 152 layers depth-wise. These layers are distributed across four convolutional blocks as laid out in Table 4.1.

Each convolutional block in a ResNet is a sequence of convolutional layers, which are tied together with skip connections. The degradation problem of deep neural networks is a phenomenon that occurs where an increase in depth leads to the counter-intuitive effect of an increase in evaluation error. This effect is countered in the ResNet architecture by combining the outputs of every few layers with the so-called "skip-connections", as illustrated in Fig. 4.1a. For the shallower versions of the network (ResNet-18 and ResNet-34), skip-connections are placed between every two layers. These layers are both convolutions with a kernel size of 3×3. For the other iterations of the architecture, which have more layers (50, 101, 152), these 3×3 convolutions become increasingly expensive to perform. To overcome this problem, a *bottleneck* structure is adopted, which uses only one 3×3 convolution positioned between two 1×1 convolutions. In these cases, the residual con-

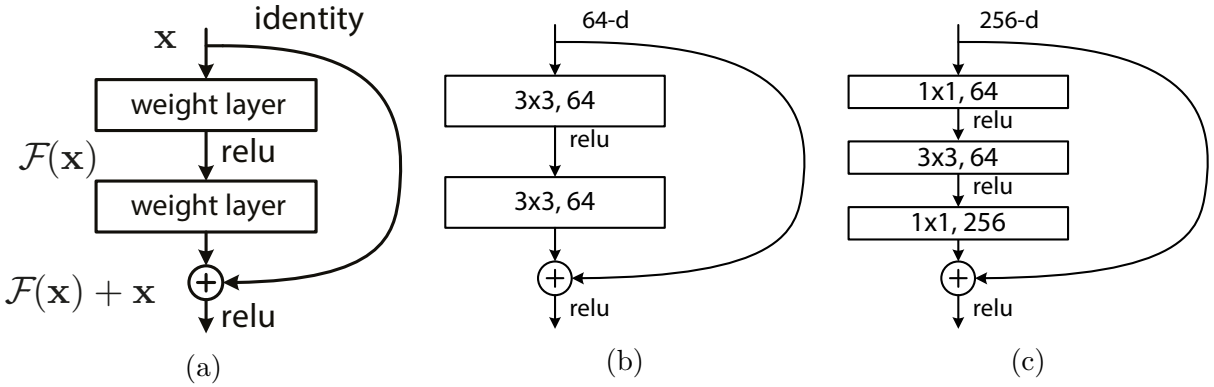


Figure 4.1: Residual skip connections used in convolutional blocks of the ResNet architecture. (a) illustrates the concept of skip connections. The output (x or *identity*) of a previous layer block is carried over to the output of the following layer block ($F(x)$) to be combined by an addition operation ($F(x) + x$). (b) and (c) illustrate the application of skip connections in regular convolutional blocks and bottleneck blocks respectively. Figures taken from (HE *et al.*, 2016).

nections are performed across every three layers. Figures 4.1b and 4.1c illustrate these structures respectively. The dimensions of each convolutional layer for every ResNet are presented in Table 4.1.Finally, the output layer of a ResNet is the sequential application of an average pooling operation to merge the feature maps on the channel dimension, a fully connected layer of output dimension 1000, and a softmax operation. The 1000-d output is chosen to match the network's output to the characteristics of the popular image classification ImageNet 2012 (RUSSAKOVSKY *et al.*, 2015) data set, which consists of 1000 classes.

4.1.2 Building ARes-14

Bello *et al.* (2019) experimented with attention augmentation on ResNet architectures on the 34, 50, 101, and 152 layers versions by using AAConv layers on every 3×3 of the last three stages of the architecture where the activation maps have height and width of 28×28, 14×14 and 7×7 (as seen in Table 4.1). In this section, we explore their design decisions and what changes we made when conceiving our version of an attention-augmented residual network.

As mentioned previously, gaze estimation does not need very deep networks to perform well. Therefore we chose the ResNet-18 version as a starting point, replacing every convolutional layer in each of the four blocks with AAConv layers. Contrary to what is done in (BELLO *et al.*, 2019), our architecture also adopts attention augmentation on the first block since our task (gaze estimation) is heavily reliant on geometric cues, which are usually in the features obtained from the first layers of a convolutional network. The only regular convolution remaining in our model is the input stem, which we maintain identical to ResNets.

With attention augmentation introduced in all 18 layers, the computational cost of

# Conv. blocks	# Layers	Params (M)	FLOPs (G)
4	18	16.94	3.26
3	**14**	**4.35**	**1.23**

Table 4.2: Comparison between the ARes networks with 4 and 3 convolutional blocks. We assess network complexity by comparing the number of trainable parameters (**M**illions) and approximate floating operations (FLOPs, in billions (G)).

training increased significantly, to the point where using large enough batch sizes for training was not possible due to hardware constraints. Using small batch sizes can be prejudicial to deep learning networks since it makes popular bias reduction techniques such as *batch normalization* (IOFFE; SZEGEDY, 2015) useless and favorable to overfitting. Several different combinations of regular and attention-augmented networks were experimented with, and we ultimately decided to drastically reduce the number of parameters of the network by cutting the last convolutional block entirely, reducing the overall depth to 14 layers while keeping the entire network as attention-augmented (with exception of the input layer). Table 4.2 shows a comparison between the number of trainable parameters and approximated floating point operations (FLOPs) before and after removing the last convolutional block. The number of trainable parameters directly relates to the memory cost and complexity of the network, while the FLOPs represent the number of multiplication and addition operations performed during a forward pass of the network. Our modified version presents a significant decrease of nearly 75% fewer parameters and approximately 60% less floating-point operations when compared with the 18-layers variant.

Each convolution and AAConv layer is followed by a batch normalization and activation (ReLU) operation. There are two hyper-parameters associated with AAConvs, the ratio between attention channels and output filters (k) and the ratio between the key depth and output filters (v). We keep both fixed to 0.25 for every self-attention augmented convolution as stipulated by (BELLO *et al.*, 2019) for use with their AA-ResNet-34 since other values tend to cause instability in the training procedure. Unless otherwise specified, the number of attention heads, Nh, is fixed to 8. Further experimentation on the effects of using smaller values for Nh are presented in Chapter 5. The output layer consists of a global average pooling layer to squeeze the feature maps over the channel dimension, to output a 1×256 feature vector.

Figure 4.2 depicts the final ARes-14 architecture. It is worth noting that like ResNets, ARes-14 is application-agnostic, being capable of being adapted to fit any task solvable by the use of convolutional neural networks. We emphasize, however, that they present a natural affinity for tasks that can perform well with fewer convolutional layers and may benefit from spatial attention cues.

4.2 ARES-GAZE: A FRAMEWORK FOR GAZE ESTIMATION

We propose a fairly conventional setup: A two-stemmed network where each branch is an instance of ARes-14, and the extracted features are joined by a shared prediction layer,

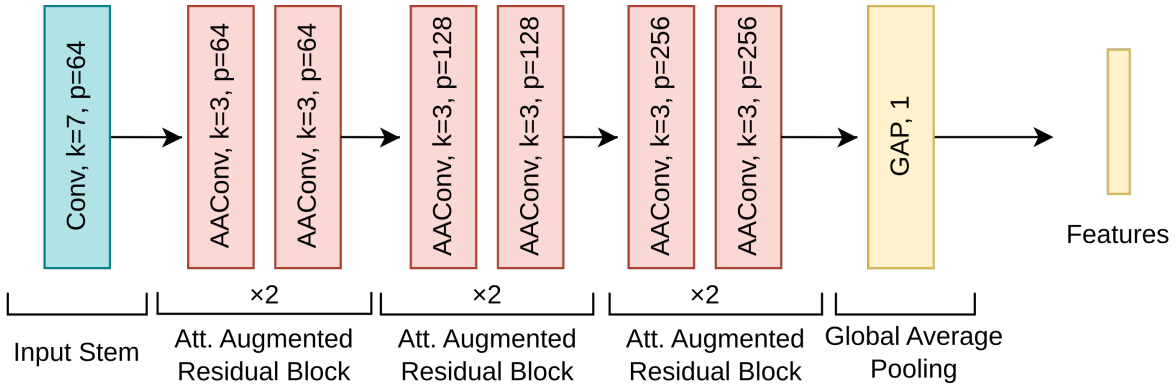


Figure 4.2: ARes-14: Self-attention augmented ResNet with 14 layers. All convolutions in residual blocks are augmented with self-attention, while the input stem remains with conventional convolutions.

as shown in Fig. 4.3.

4.2.1 Inputs

Many works have used multi-input frameworks in appearance-based gaze estimation (CHENG *et al.*, 2020; FISCHER; CHANG; DEMIRIS, 2018; CHEN; SHI, 2020; PALMERO *et al.*, 2018; ZHU; DENG, 2017). The use of multiple inputs in most modern gaze estimation methods is attributed to the fact that the gaze direction of a subject relies heavily on more than one factor (eyes, head pose, and location, distance). For the ARes-Gaze framework, we elected to use RGB-face images (normalized for pose and distance) and grayscale histogram-normalized eye images as our inputs, as is the standard for multi-input convolutional networks for gaze estimation.

To extract information from the eye images, some published methods with similar topologies use two networks (one for each eye) (CHEN; SHI, 2020; FISCHER; CHANG; DEMIRIS, 2018) or a single network with shared weights (making separate passes for each input) (CHENG *et al.*, 2020). Here we employed a single-pass, single-network strategy for the eye branch by stacking the left- and right-eye regions, creating a $1:1$ ratio square input. Intuitively, this should reduce the network complexity by using a single network (thus reducing the number of parameters) and reduce the execution time by only needing to make a single pass for both eyes. We further justify and study the practical implications of the use of this method in comparison with the other mentioned works in Section 5.2.2.1.

4.2.2 Feature extraction

We use identical backbones to extract features from the face images and the eye patches. Theoretically, the framework is agnostic w.r.t the feature extractor used, but the goal here is to exploit ARes-14 and its spatial awareness potential to improve upon regular CNNs.

In order to determine if and how relative self-attention has a positive effect on gaze

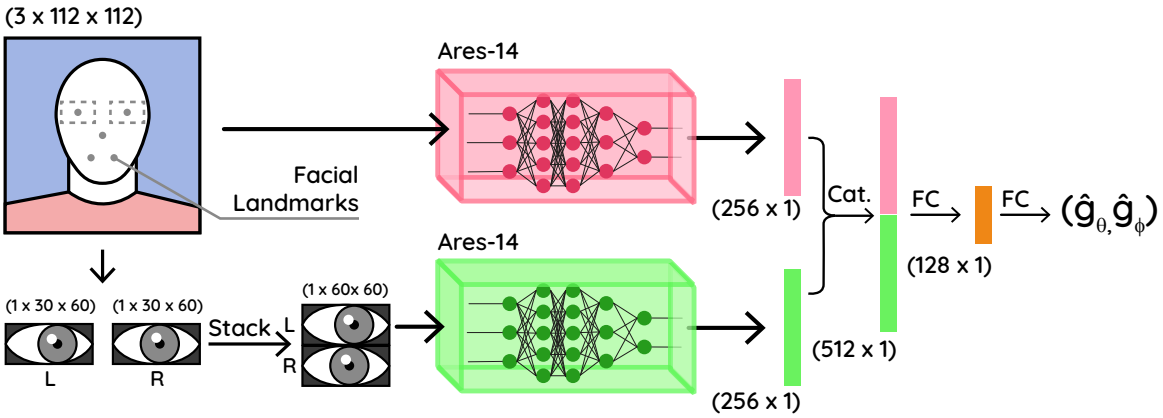


Figure 4.3: ARes-gaze framework. Face- and eye-patches are extracted and separately normalized from the source image. The normalized inputs are then sent through twin ARes-14 backbones. The resulting features are concatenated and passed through a prediction stage consisting of two fully-connected layers.

estimation, we trained multiple models following the proposed topology, exchanging the self-attention augmented feature extractor in each branch (ARes-14) for a baseline regular convolutional one (ResNet). In the service of fairness, we remove the final convolutional block from the ResNet-18 network used for comparison in the same way that it is done for ARes-14. This baseline is referred as ResNet-14.

To further investigate whether self-attention has an equal impact on the different input types of our task, we also trained baseline and self-attention augmented single-branch versions of the framework, with only face or eye inputs to observe the isolated effects of attention in each type of input. The results of these experiments are presented in Table 5.3 in Section 5.2.2.3.

4.2.3 Output layer

As previously mentioned, the third convolutional block in ResNet architectures outputs feature maps of size 256 for the channel dimension. Given that in our ARes-14 architecture this is our last convolutional block, we apply global average pooling directly on the output to obtain a feature vector of dimensions 1×256. The outputs from both branches of the ARes-Gaze framework are joined by a simple concatenation operation giving equal weight to the features extracted from each input. The features on the resulting 512-d vector are combined by a fully connected layer to form a 128-d embedding vector that should contain all the information regarding the gaze direction (see the orange-colored block in Fig. 4.3). The final prediction is obtained by another Fully Connected layer that outputs the pitch ($\hat{\mathbf{g}}_\theta$) and yaw ($\hat{\mathbf{g}}_\phi$) angles.

4.2.4 Implementation Details

The code was written with the PyTorch (PASZKE *et al.*, 2019) deep learning framework. All the models were trained for 120 epochs with a batch size of 48 on an HPC cluster equipped with 8 NVidia V100 GPUs [1]. The high computational overhead of training attention-based methods is prohibitive with regards to the batch size, and this needs to be taken into account when choosing hyper-parameters.

We used the stochastic gradient descent (SGD) (BOTTOU, 2010) solver with a momentum equal to 0.9. The training of the self-attention models was sometimes unstable, and vulnerable to the vanishing gradient problem in the early iterations of training. To counteract this, we employed a warm-up schedule for the learning rate, linearly increasing it every epoch for 5% of the total epochs until it reaches a value of 0.128. For the rest of the training, we applied cosine annealing (LOSHCHILOV; HUTTER, 2016) to gradually decrease the learning rate and avoid over-fitting. We empirically found that a weight decay parameter of 0.0003 was also helpful in preventing over-fitting. The training loss was calculated using the smooth $L1$ cost function between the ground-truth and predicted values. Proposed as a regression loss for the Fast R-CNN network (GIRSHICK, 2015), the smooth $L1$ is a "middle-ground" between the $L1$ and $L2$ cost functions, being more robust than $L1$ loss but less sensitive to outliers than the $L2$ loss (which can cause gradients to explode during training). The smooth $L1$ function is defined as follows:

$$\text{smooth}_{L1}(x, y) = \begin{cases} 0.5(x-y)^2, & \text{if } |x-y| < 1 \\ |x-y| - 0.5, & \text{otherwise}\,, \end{cases} \tag{4.1}$$

where x and y are the ground-truth and predicted vectors.

4.3 CLOSURE

This chapter explained the structural details, design choices, and practical implementation of the ARes-Gaze framework. As the main component of our proposal to use self-attention in appearance-based gaze estimation, the work hinges on properly evaluating the effectiveness of the framework. The next chapter explains the methodology adopted and the experiments performed to investigate the effects of self-attention on this particular task of gaze estimation.

[1] courtesy of the National Laboratory for Scientific Computing (LNCC/MCTI, Brazil) URL: http://sdumont.lncc.br.

CHAPTER 5

EXPERIMENTAL ANALYSIS

Contents

In this chapter, we justify and report the results from the experiments made to prove the validity of our proposal. First, we characterize the data sets used to evaluate and compare our methods to the current state-of-the-art. We also explain the pre-processing pipeline used to normalize the data in all experiments. Then, we present the methodology and results of ablation studies made to quantitatively evaluate the impact of self-attention in our model by comparing it with a fully convolutional baseline. Finally, we present additional studies with regards to important external factors specific to the task of appearance-based gaze estimation (head pose angle and illumination conditions).

5.1 EVALUATION SETUP

5.1.1 Training data

With appearance-based models (more specifically, deep learning-based approaches) becoming the new standard in gaze estimation, the need for data to train those models pushed researchers to build and release increasingly larger and better data sets. Out of all the data sets mentioned in Section 3.1.2, we elected two of the most used publicly-available data sets to perform our evaluations on, the MPIIFaceGaze (ZHANG *et al.*, 2015, 2017) which is a modified version of the MPIIGaze data set, and the EyeDiap (MORA; MONAY; ODOBEZ, 2014) data set.

Figure 5.1: Data samples from the EyeDiap (MORA; MONAY; ODOBEZ, 2014) (top row) and MPIIFaceGaze (ZHANG *et al.*, 2015) (bottom row) data sets. The samples on the top row were normalized by following the procedure described in Section 5.1.2. The bottom samples were taken from the already-normalized MPIIFaceGaze data set (ZHANG *et al.*, 2017).

We chose these two data sets due to three important factors that separate them from the other available data sets for gaze estimation described in Section 3.1.2: first, they are the most commonly used for experimentation among the published works we judge similar to ours (mentioned in Table 3.1). Since we will use those models as a comparison baseline for the evaluation of the accuracy of our proposal, it is necessary to perform our evaluations on the same data. Second, the two data sets provide large amounts of data while still presenting challenges w.r.t. various scenarios and illumination conditions (in the case of MPIIFaceGaze) and wide ranges of head pose angles (in the case of EyeDiap). Finally, these data sets are not limited to a specific application niche, such as mobile devices in the case of TabletGaze (HUANG; VEERARAGHAVAN; SABHARWAL, 2017) and GazeCapture (KRAFKA *et al.*, 2016).

Figure 5.1 shows samples of training data from both selected data sets, and Table 5.1 summarizes the most relevant characteristics from them.

The MPIIFaceGaze data set. The original MPIIGaze data released to the public contained only crops of the eye regions of the subjects, with the rest of the image being blacked out (Fig. 5.2). With the release of the Spatial Weights CNN (ZHANG *et al.*, 2017), which performed gaze estimation making use of full-face images, the authors also made public a modified version of the data set entitled MPIIFaceGaze. This new version contains 3,000 full-face already normalized images for each subject. Figures 5.3c and 5.3f show the gaze data distribution for the MPIIFaceGaze data set.

The EyeDiap data set. The EyeDiap data set is split into three modalities, as explained in Section 3.1.2: discrete screen target, continuous screen target, and 3D floating target. In our experiments, we used only the modalities where the gaze target was projected

Characteristics	**Data sets**	
	MPIIFaceGaze	EyeDiap
Size	45,000 images	94 videos
Image type	RGB	RGB-D
Subjects	15	16
Subject distance	40 - 60cm	80 - 120cm
Normalized	✓	–
Head-pose annotation	✓	✓
Extreme-head pose	–	✓
Extreme-lighting variation	✓	–
Eye-position annotation	✓	✓

Table 5.1: Comparison of the relevant characteristics of the MPIIFaceGaze (ZHANG *et al.*, 2015, 2017) and EyeDiap (MORA; MONAY; ODOBEZ, 2014) data sets used in the experiments.

onto the screen (continuous and discrete). Unfortunately, in the floating target-sessions, the small ball would sometimes occlude the subject's face, hindering the use of full-face approaches. Two of the subjects are only present in floating target sessions, so we are left with a total of 14 subjects and 56 videos. There are two different versions of each session: One where the subject's head is stationary with the target being pursued with eye movements only, and another where the subject also moves the head.

Figures 5.3a, 5.3b, 5.3d, and 5.3e show the gaze and head-pose data distribution for the EyeDiap data set grouped by whether the subject was instructed to move its head (static or mobile).

As mentioned, the data is originally in video format. To compose our training and evaluation data subsets, we sampled every 5^th frame from the recordings. We cleaned the resulting frames by excluding those with missing annotations for the head pose, screen target location, or eye position. Annotations for invalid frames because the subjects have their eyes closed (blinking) or are distracted (looking away from the target) are available for some of the data set's sessions, but not all. For those sessions without such annotations, we manually parsed extracted frames, removing those considered invalid. In the end, approximately 44.000 examples remained to carry out with the leave-one-person-out cross-validation.

5.1.2 Data normalization

As mentioned previously in Section 4.1, relatively shallow convolutional neural networks are sufficient for the gaze estimation task, particularly when comprehensive pre-processing pipelines are in place to normalize the data. In this section, we describe the normalization procedures adopted in our experiments for the face images and the eye images our proposal uses as inputs.

Face images. For the full-face images, we follow a procedure similar to that presented in

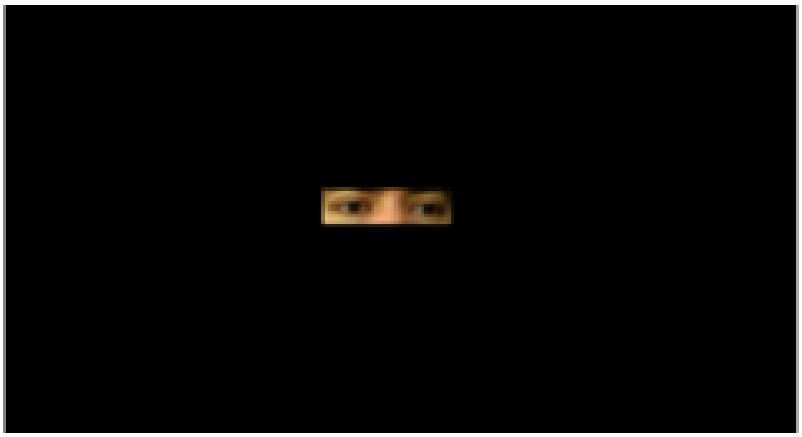


Figure 5.2: Data sample for the MPIIGaze data set as made available to the public. The image captured by the camera is left with only a small region around the eyes is visible, with the rest of the image being occluded.

(SUGANO; MATSUSHITA; SATO, 2014). An affine transformation is applied to rotate the input as to cancel out the roll-axis angle of the head and simultaneously scale it to the desired size. We used RGB images with an input size of 112×112 pixels.

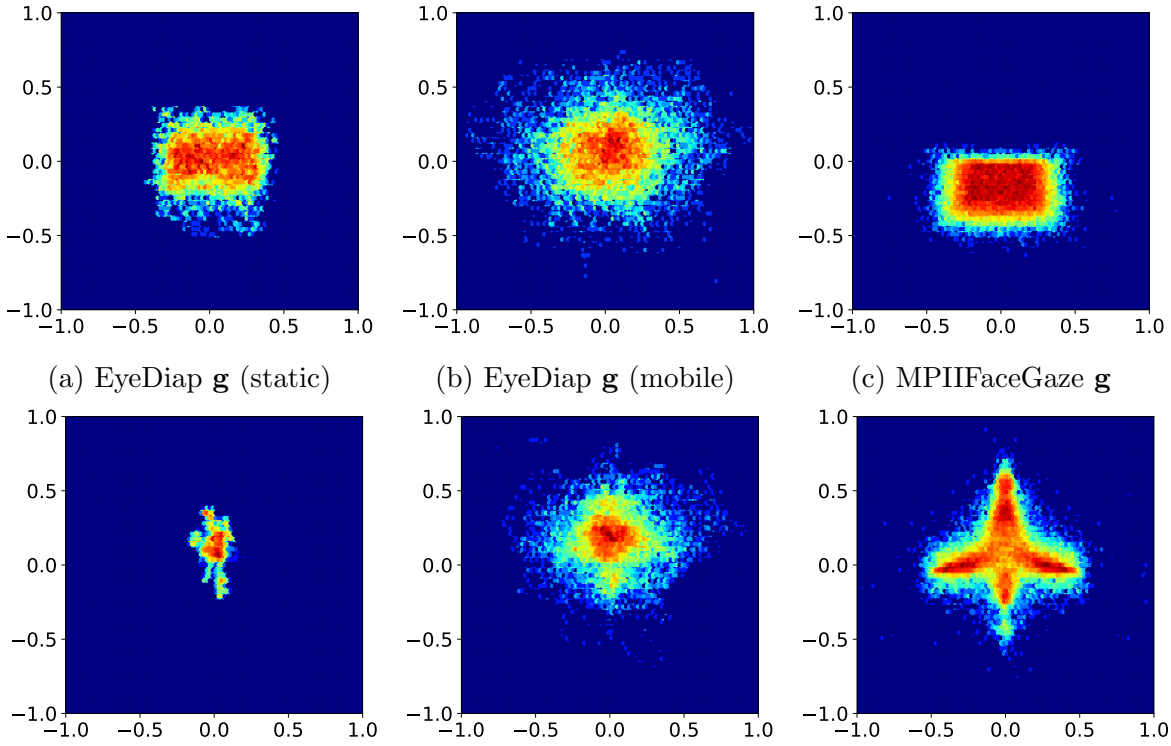
The effect of the affine transformation is that relevant facial features are always in the same regions on the input, making it easier for the network to recognize patterns in important regions. This also standardizes the distance from the face to the virtual camera reducing the complexity of the parameters the network must learn. In our experiments, this procedure is only applied on the EyeDiap (MORA; MONAY; ODOBEZ, 2014) data set since the MPIIFaceGaze data set (ZHANG *et al.*, 2017) is already normalized. Figure 5.4 illustrates these procedures as a pipeline.

The affine transformation is the matrix M, which is defined as $M = [R\ T]$, where R is the rotation component that normalizes the roll-axis rotation, and T is the translation component that ensures the relevant patch from the original frame is in the normalized image. To build R, we need the parameters α (rotation) and S (scale). We estimate the angle of the line between the subject's eyes, and use that value as the parameter α, which can be obtained with the help of a facial landmark detector, although here we use the annotated position provided by the Eyediap data set. The scale parameter, S, controls the distance to the subject in the image, abstracted as the size of the face. For a squared input, we defined that the distance, d, between the left- and right-eye centers should be 40% of the image width. Given that the face should be vertically centered, this gives us left- and right-eye centers to be $(0.7, 0.35)$ and $(0.3, 0.35)$ on a normalized 0 to 1 scale relative to the input dimensions.

S can then be given as

$$S = \frac{Z * d}{D}, \tag{5.1}$$

where Z is the final image size in pixels (considering square images, in our case $Z = 112$), and D is the original distance between the subject's eye center.



(a) EyeDiap **g** (static) (b) EyeDiap **g** (mobile) (c) MPIIFaceGaze **g**

(d) EyeDiap **h** (static) (e) Eyediap **h** (mobile) (f) MPIIFaceGaze **h**

Figure 5.3: Gaze ground-truth (**g**) and normalized head pose (**h**) distribution on the MPIIFaceGaze and EyeDiaps data sets. The latter is split into static or mobile according the subject's head movement. Angle values are displayed in radians.

With the angle and scale parameters in hands, we can then build R as follows

$$R = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}, \tag{5.2}$$

where a = S * cos(α), b = S * sin(α).

T is defined as

$$T = \begin{bmatrix} (1-a) * GO.x - b * GO.y \\ b * GO.x + (1-a) * GO.y \end{bmatrix} + \begin{bmatrix} tX - GO.x \\ tY - GO.y \end{bmatrix}, \tag{5.3}$$

where tX = Z * 0.5 , tY = Z * 0.35, and GO is the coordinates of the gaze origin vector (here, defined as the middle point between the left- and right-eye centers). The tX and tY are the correction factors that enable re-positioning the gaze vector in the normalized image.

Eye images. The eye patches are cropped from the normalized face. To do so, we need prior knowledge of the location of the eyes.

EyeDiap provides annotations for five facial landmarks, two of them being the centers of the left and right eyes. The eyes are obtained from the normalized face by cropping

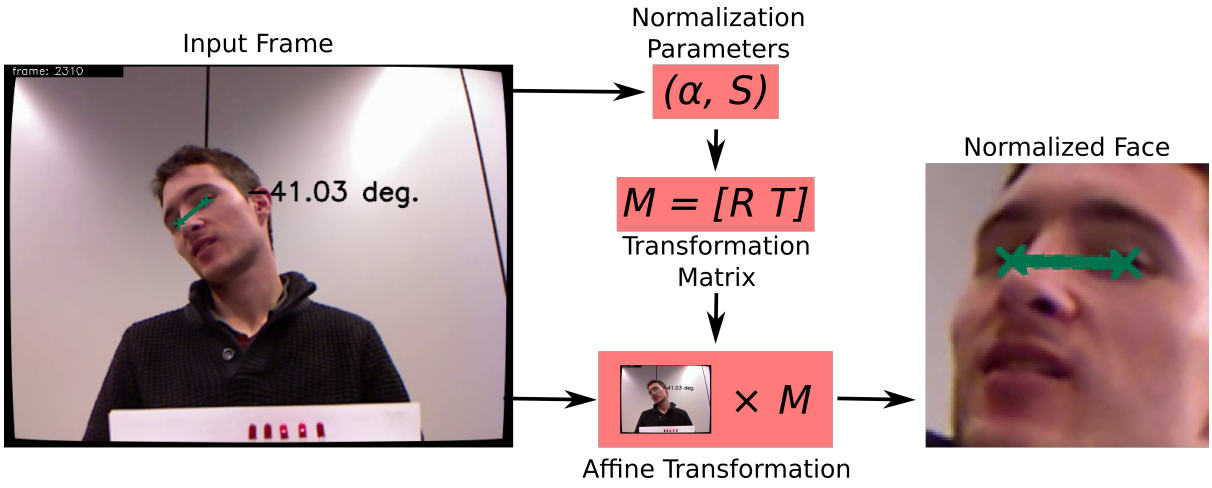


Figure 5.4: Normalization procedure on a sample frame from the EyeDiap data set. The line drawn between the subject's eyes is used to determine the α and S parameters needed to build the transformation matrix M.

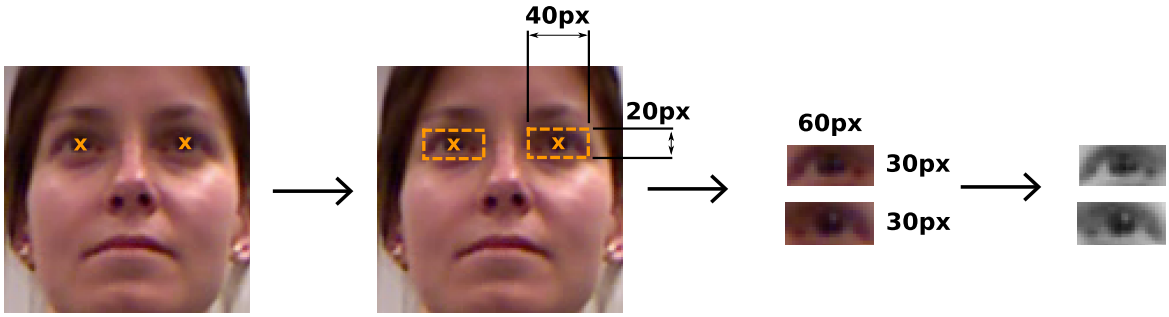


Figure 5.5: Eye normalization procedure on a sample frame from the EyeDiap data set. The facial landmarks are used to crop a 20×40 region around the center of the eye, which is then cropped from the face, resized to 30×60, converted to grayscale and histogram normalized.

a rectangular region of size 20×40 pixels around the eye center landmarks. The resulting crops are converted to grayscale, histogram-normalized, and resized to the input dimensions, which we established as 30×60 pixels.

The eye cropping and normalization steps are carried out for both data sets and are illustrated in Fig. 5.5.

5.2 EXPERIMENTAL RESULTS

To properly evaluate our proposed framework, a group of experiments was carried out and divided into two main parts: First, a set of ablation studies were performed to assess the impact of the self-attention augmentation modules on individual aspects of the ARes-gaze architecture. The main goal of this part is to better understand the optimal conditions in which to apply AAConvs in our framework. In this section, the studies were

namely: The input modalities of the eye images, the performance of ARes-14 *versus* a regular ResNet14, and the number of attention heads for the self-attention augmented layers. In the second part, external factors that directly impact the performance of gaze estimation were analyzed to explore how our proposed framework can deal with them. Here we perform two experiments to explore how the robustness to head-pose variation and illumination conditions of the input image is affected by using self-attention.

5.2.1 Evaluation methodology

We adopt a leave-one-out cross-validation strategy performed across the subjects from each data set. This strategy allows for better reproducibility of the results and reduces the effects of subject bias in the data. Considering the characteristics of the data sets used in the experiments (see Table 5.1), N models were trained where N is the number of available subjects in each data set. For each trained model, a different subject is held out and used for testing. The final result is the average of the evaluations of all models. On the EyeDiap data set, for example, the final scores are the average performance of the 14 trained models on the held-out subject, each time. Similarly, on the MPIIFaceGaze data set, the final result is obtained by averaging the accuracy scores from 15 trained models.

5.2.2 Ablation studies

The first evaluation we present here is grounded on different input models for the eye branch. The goal was to compare our proposed single-branch, single-pass vertical stacking scheme (see Section 4.2.1) with other strategies adopted by similar methods. Subsequently, we present studies on the effect of self-attention augmentation on different inputs (face and eyes) and network schemes. The aim is to understand how and where self-attention is effective on the task of gaze estimation, and how it ultimately impacts the overall performance of the framework. By evaluating isolated portions of the proposed framework with and without self-attention augmentation, these experiments are useful in generating insights on how ARes-14 can be best applied, guiding future researches.

Finally, we evaluate the effect of choosing different numbers of attention heads for ARes-14. The multi-headed attention mechanism can present significant computational overhead, so an investigation on the trade-off between the number of attention heads and evaluation error drives this choice.

5.2.2.1 Evaluating different models of the eye images The inputs to the eye branch of our network are images of both eyes from the subject. There are two ways that models with network topologies similar to ours usually adopt to feed the eye images to the CNN. In (CHENG *et al.*, 2020), the eyes are passed through a single network with shared weights one at a time. We deem this approach to be sub-optimal in that it requires two passes to process both eyes. This is undesirable especially since we already use multiple inputs in the network (face and eyes) and we want to reduce the computational overhead as much as possible due to the attention layers used.

Another more common approach is to have a dedicated network branch for each eye,

Model type	Average angular error		# Params (M)	FLOPs (M)
	MPIIFaceGaze	EyeDiap		
SEI (Ours)	**5.40°**	**7.27°**	**2.810**	**414**
DP-SW	5.54°	7.42°	2.842	422
DP-TB	5.45°	7.36°	5.619	422

Table 5.2: Results on different input models of the eye images. The evaluated parameters considered are: Average angular error on the EyeDiap and MPIIFaceGaze data sets, number of trainable parameters (**M**illions), and approximate floating operations (FLOPs, also in **M**illions) for the three evaluated input models.

with identical topologies and layers but distinct weights (CHEN; SHI, 2020; FISCHER; CHANG; DEMIRIS, 2018). This methodology has the same problem as the shared weights approach, with the network needing to perform two forward passes to process the eyes. Additionally, this approach has double the number of trainable parameters, making it even less attractive for our use case.

To address both of the concerns presented, we propose the vertical stacking of eye images to obtain a single 1:1 input image that can be processed in a single pass by a single network branch. To evaluate whether this proposal is valid, we compare it to the other mentioned modalities considering the following characteristics: the number of trainable network parameters, approximated floating-point operations (FLOPs), and average angular evaluation error.

- Stacked-eyes input (SEI)
- Double Pass - Shared Weights (DP-SW)
- Double Pass - Twin Branches (DP-TB)

The three input modalities are illustrated in Fig. 5.6 to better highlight they differences.

As summarized in Table 5.2, although there is arguably only a small difference in the average angular error, the stacked-input model performed better than the other ones on both data sets. The stacked-input model also presents roughly the same number of trainable parameters of the shared-weights variety and a significantly lower number when compared to the twin-branch network. These results further validate the adoption of the stacked-eye for the ARes-Gaze framework and all subsequent evaluations.

5.2.2.2 ARes-14 evaluation Intending to gauge the effect of self-attention augmentation in multiple stages of ARes-gaze, we evaluated and compared multiple models based on the ARes-14 architecture. First, to see how attention affects different types of input, we trained single-branch networks with and without self-attention augmentation. We evaluate different versions where the inputs were only eye images or only face images. Second, we evaluate the entire ARes-Gaze framework, comparing models where we switch between ResNet-14 and ARes-14 backbones for each input branch. The goal is to explore

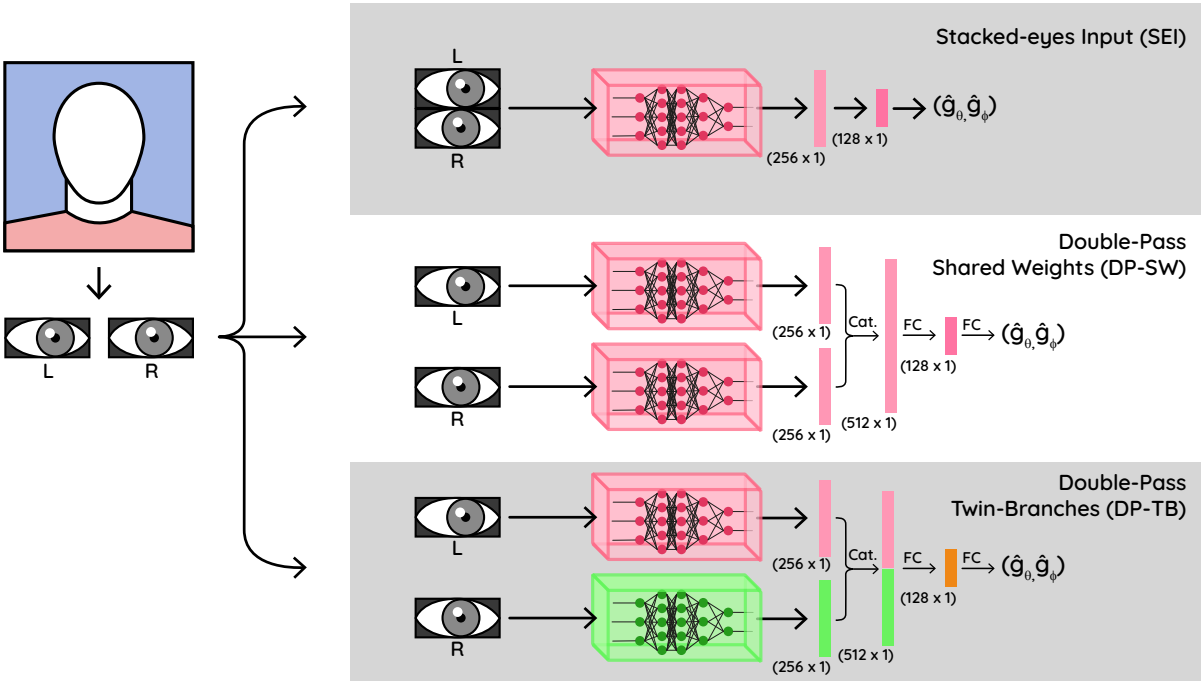


Figure 5.6: Visual comparison of the three evaluated eye modalities. The top row shows how our proposed approach only needs a single pass through the network to produce results. In the middle row, both networks have the same color to indicate they have shared weights. As such, it is necessary to perform one pass at a time through the same backbones. The different colors used in the networks in the bottom row mean that they don't share their weights.

the contrast between fully convolutional features and self-attention augmented features for gaze estimation.

The results for the leave-one-out cross-validation for each model are laid out in Table 5.3. The network types are single regular, single attention, or both regular/attention branches. For the single-branch networks (with either only face or only eyes as inputs), we observe a drop of more than 17% on the average angular error on the EyeDiap data set when using self-attention augmented convolutions. When compared with its regular convolutional form, ARes-gaze reduces the average error by 6.05% on the MPIIFaceGaze data set and by 8.4% on EyeDiap.

5.2.2.3 Determining the number of attention heads In all evaluations reported in the paper which proposed the structure of Attention Augmented Convolutions (BELLO *et al.*, 2019), all accuracy gains presented are on architectures using a fixed number of attention-heads, specifically $Nh = 8$. In this section, we evaluate ARes-gaze considering other values of Nh to isolate and gauge the effect of the number of parallel attention heads on the final evaluation results.

Table 5.4 shows the average angular errors found on the MPIIFaceGaze and EyeDiap data sets. Notably, for the MPIIFaceGaze data set, when using less than 4 attention-

Network type	Input		Data set	
	Eyes	Face	MPIIFaceGaze	EyeDiap
Regular	■		5.40°	7.27°
Attention	■		5.33°	6.02°
Regular		■	4.71°	7.42°
Attention		■	4.46°	6.10
Regular	■	■	4.46°	6.09°
Regular Attention	■	■	4.42°	5.81°
Regular Attention	■	■	4.52°	5.84°
Attention	■	■	**4.19°**	**5.58°**

Table 5.3: Results of attention-augmented versus regular convolutional layers on the backbones of ARes-gaze. Best results are highlighted.

Method	MPIIFaceGaze	EyeDiap
Baseline	4.46°	6.09°
ARes-gaze (Nh=2)	4.93°	5.98
ARes-gaze (Nh=4)	4.36°	5.99
ARes-gaze (Nh=8)	**4.19°**	**5.58°**

Table 5.4: Results of average angular errors on different numbers of attention-heads per attention layer. Best results are highlighted.

heads, the ARes-gaze architecture performs worse than the purely convolutional baseline, with the evaluation error proportionally decreasing with the increase of attention-heads. On the EyeDiap data set, the results follow the same tendency with $Nh = 2$ and $Nh = 4$, which are only marginally better than the baseline network. In both data sets, there is a sudden and significant improvement in the results when $Nh = 8$. Numbers greater than 8 were not experimented with due to memory constraints in the available hardware.

5.3 COMPARISON WITH OTHER APPEARANCE-BASED METHODS

We selected six appearance-based methods that take as input either full-face images or a combination of full-face images and other inputs. All these methods output a single-gaze vector with origin in the center of the face or in the middle-point of the eye. The selected methods were: the iTracker in its original form (KRAFKA *et al.*, 2016) and with AlexNet backbone (ZHANG *et al.*, 2017), the CNN with spatial-weights mechanism (ZHANG *et al.*, 2017), RT-Gene (a version of 4 ensembles with the best reported results) (FISCHER; CHANG; DEMIRIS, 2018), the CNN with CNN dilated convolutions (CHEN; SHI, 2020), and the eye-asymmetry based FAR-Net (CHENG *et al.*, 2020). These are approaches we consider similar to ours, which were compared over the average 3D-angular error on the chosen data sets. Except for RT-Gene and iTracker (AlexNet), which do not report

Method	MPIIFaceGaze	EyeDiap
iTracker (KRAFKA *et al.*, 2016)	6.2°	8.3°
iTracker (AlexNet) (KRAFKA *et al.*, 2016; ZHANG *et al.*, 2017)	5.6°	–
Spatial Weights CNN (ZHANG *et al.*, 2017)	4.8°	6.0°
RT-Gene (4 Ensemble) (FISCHER; CHANG; DEMIRIS, 2018)	4.3°	–
Dilated CNN (CHEN; SHI, 2020)	4.5°	**5.4°**
FAR-Net (CHENG *et al.*, 2020)	4.3°	5.7°
Baseline	4.5°	6.1°
ARes-gaze (Nh = 8)	**4.2°**	5.6°

Table 5.5: Results of average angular error compared with other appearance-based methods. Best results are highlighted.

evaluations on the EyeDiap data set, all compared methods use the same or a similar protocol to extract data from the videos, as described in Section 5.1.1.

The results are reported by considering two versions of our architecture: The full ARes-gaze and ARes-gaze without self-attention augmentation (baseline).

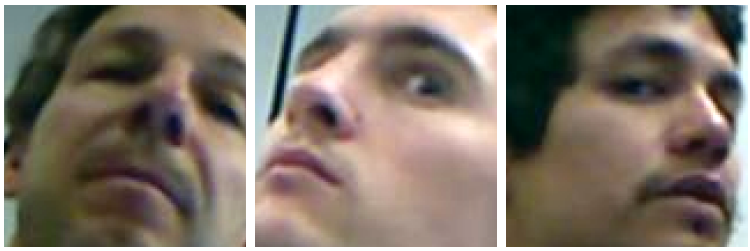
When compared to the other methods, our ARes-gaze framework with twin ARes-14 backbones reached state-of-the-art results on the MPIIFaceGaze data set, and the second-best place on the EyeDiap data set, being only 0.2 degrees behind the Dilated CNN (CHEN; SHI, 2020) (see Table 5.5). It is worth noting that no other method was able to have superior results on both data sets.

5.4 EVALUATION OF THE IMPACT OF EXTERNAL FACTORS ON THE PROPOSED APPROACH

In in-the-wild gaze estimation applications, the recording conditions are sub-optimal in that the subject's head pose and external illumination conditions are to be considered unconstrained. As such, these are highly relevant factors to be considered when proposing new approaches for gaze estimation. As discussed in Section 5.1.2, we applied color-level normalization, and we enforced roll-angle normalization during training and inference to reduce the complexity of our model when the data set does not offer already normalized images. Regardless, edge cases of extreme conditions are still challenging, and there are still the pitch and yaw head pose angles to be concerned about. To see if and how self-attention augmentation in convolutions affects robustness to these factors, we evaluated both our completely attention-augmented model and the regular convolutional baseline in isolated scenarios.

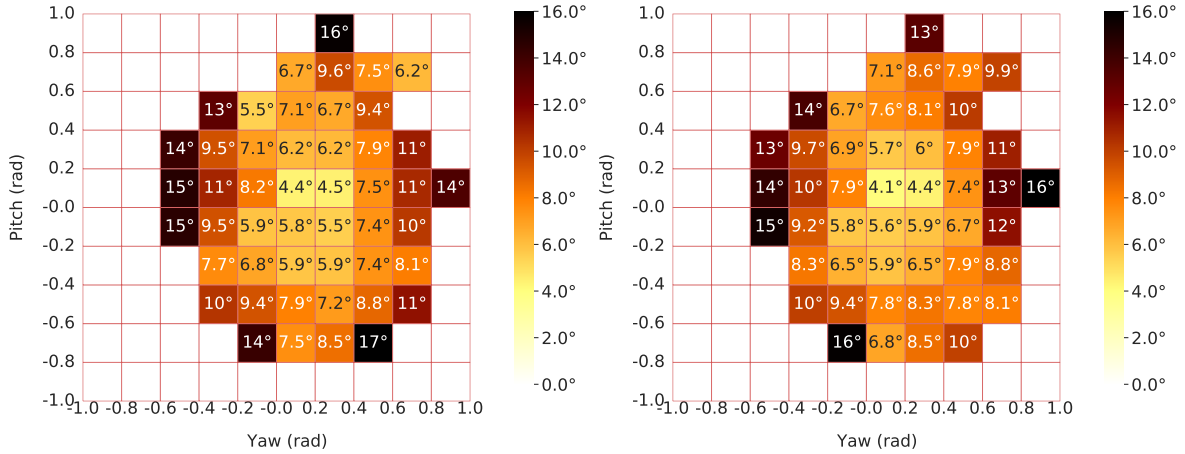
5.4.1 Head pose

To evaluate how each model performs to the subject's head pose, we used the EyeDiap data set due to its larger variety (refer to Fig. 5.3) of head pose and edge cases of extreme pose conditions (see image samples in Fig. 5.7). The data set provides annotations for the subject's head pose angles. We used this data to correlate every sample prediction error during the leave-one-out cross-validation to the subject's head pose. To visualize



(a) (b) (c)

Figure 5.7: Samples with extreme head pose angles from the EyeDiap dataset.



(a) Baseline convolutional model (b) attention-augmented model

Figure 5.8: Distribution of mean angular error of baseline and attention-augmented models across head poses in the EyeDiap data set.

the results, we built a 2D heat-map where the axis represents the pitch and yaw angles, and each cell represents the average error for that combination of angles. The cell values were calculated by performing 2D binning at regular intervals of 0.20 radians across the entire data set.

Figure 5.8 shows the results for both self-attention augmented and regular convolutional based architectures. The plots make it clear that there are overall gains across most of the pitch and yaw head-pose spectrum. This is further reinforced by Fig. 5.9, where we plotted the average angular error difference between the two plots presented in Fig. 5.8.

The overall decrease in average error appears mostly uniform outside of the most extreme cases. For those, we observe that the most significant gains obtained by the ARes-gaze model were in regions of extreme pitch angle (negative and positive), while the losses were in sections of high yaw angles. It should be noted that the overall magnitude of the gains in average accuracy is greater than that of eventual losses suffered by the model (as seen from the color-bar in Fig 5.9).

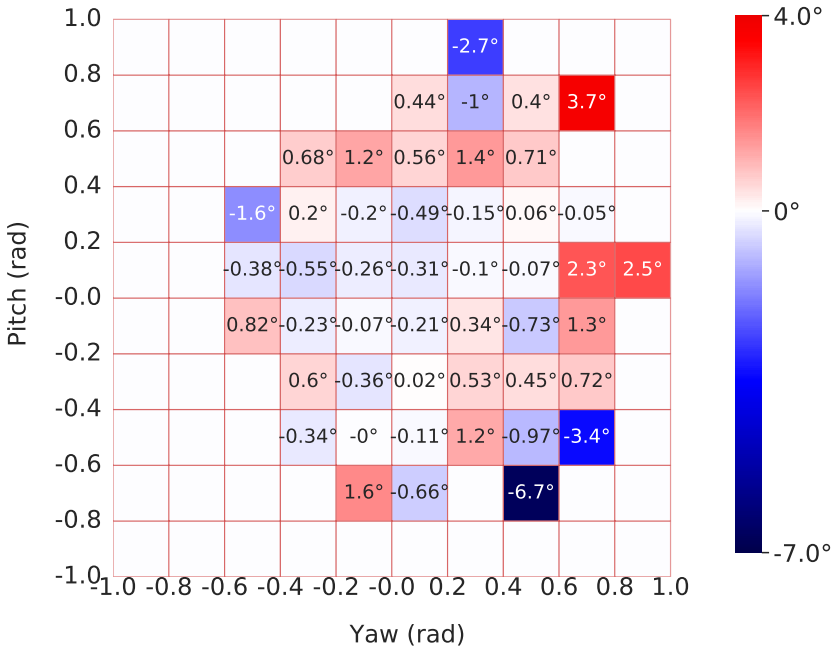


Figure 5.9: Distribution of the angle-error difference between attention-augmented and baseline models on the head-pose evaluation. Blue boxes (negative numbers) mean an improvement over the baseline model or a drop in the average angular error. Similarly, red boxes mean regions where there was an increase in the average angular error.

5.4.2 Illumination conditions

In Fig. 5.10, the illumination distributions from both the MPIIFaceGaze and EyeDiap data sets are plotted. The light level values were determined by converting the images to grayscale and averaging the pixel intensities to obtain a value between 0 (completely dark) and 255 (completely bright). The plot shows that the distribution for the MPIIFaceGaze data set is more varied when compared with EyeDiap, and also approximate a normal distribution. These are properties that should be beneficial to our evaluation, so we choose to analyze the effect of self-attention augmentation *versus* lighting on the MPIIFaceGaze data set.

To perform the evaluations, we split the 0-255 light-level range into 10 bins, and average the results across all 15 subjects with each bin. Figure 5.11 shows an overlapping evaluation of both baseline and ARes-Gaze models by light-level intervals. There is an inverse relationship between light level and angular error that behaves somewhat linearly. Additionally, the last bin, which represents extreme high-light levels (overly lit images), shows a small spike in the averaged angular error. This situation reinforces the intuitive notion that prediction models have worse accuracy on overexposed input images as well as with poorly lit ones.

To quantify the sensibility of each model to lighting conditions, we fit a regression line across the average angle error of each bin and calculated its slope (m). The closer to zero the slope is, the lower is the model's sensibility to light. This experiment shows that

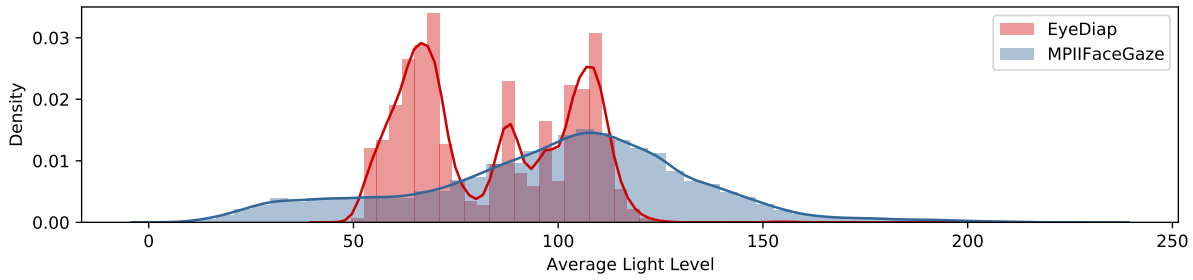


Figure 5.10: Distribution of average light intensity per sample on the MPIIFaceGaze and EyeDiap data sets.

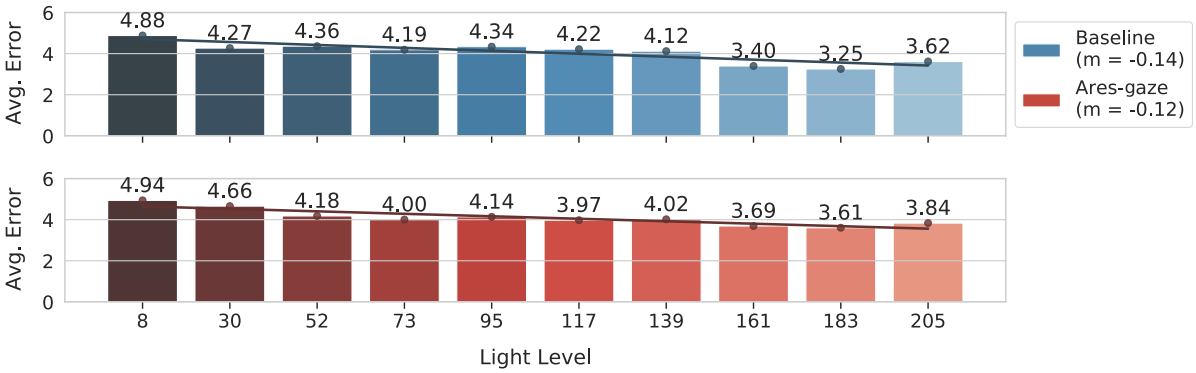


Figure 5.11: Evaluation of model accuracy *versus* lighting conditions of input images. The MPIIFaceGaze data was split into 10 bins with regards to light levels, with the X-axis showing the average level of each bin. The Y-axis is the average angular error in degrees. A regression line drawn across each model bars, with its slope value (m) being shown in the plot legend.

ARes-Gaze had a slightly smaller absolute slope inclination, although the difference was not enough to justify conclusions about its robustness to lighting conditions in comparison with the purely convolutional baseline.

5.5 CLOSURE

In this chapter, we characterized multiple experiments and their respective motivations to evaluate multiple aspects of the quantitative effects of ARes-14 and ARes-Gaze in practice. Next and final chapters, we interpret the obtained results and present our thoughts with regards to the possible future avenues of investigation for gaze estimation.

CHAPTER 6

DISCUSSION AND CONCLUSIONS

Contents

In this chapter, we contextualize our findings and attempt to answer the main questions that drove our work by interpreting the quantitative results obtained in the previous chapter. Additionally, thoughts on practical implications of the work and qualitative analysis of the attention feature maps are presented.

6.1 ON THE USE OF SELF-ATTENTION AUGMENTED CONVOLUTIONS FOR GAZE ESTIMATION

First, we evaluated the difference between using attention augmentation on eye images *versus* using attention augmentation on the entire face as inputs by training separate single-branch networks for each type of input. In these experiments, the eye images were stacked before being forwarded through the network, as reported in Section 5.2.2.1.

Intuitively, the difference between using full-face images and isolated-eye regions as inputs is the scope of the information that the network can extract. With full-face images, CNN has the chance to learn not only from the eyes themselves but also to extract head-pose information from regions such as the nose and mouth. This ability comes with the drawback of the subject's eyes having a lower resolution, thus limiting the amount of information present in their regions. In contrast, using isolated eye-patches should allow the network to extract more detailed information about the position of the pupils, making the model more sensitive to smaller changes in gaze direction. In this case, the drawback is the absence of elements that can inform the network about the subject's head pose, which we previously established has relevance to the final prediction.

Figure 6.1 shows a visual summary of the results of the single-branch networks from Table 5.3. There is a clear and consistent decrease of the average angular error in all instances when using the networks with self-attention augmented convolutions (ARes-14). As to which kind of input benefits the most from attention, on the EyeDiap data set, an error decrease of 17.19% with eyes as input *versus* 17.78% with faces can be observed. On the MPIIFaceGaze data set, the decreases were of 1.28% and 5.31%, respectively.

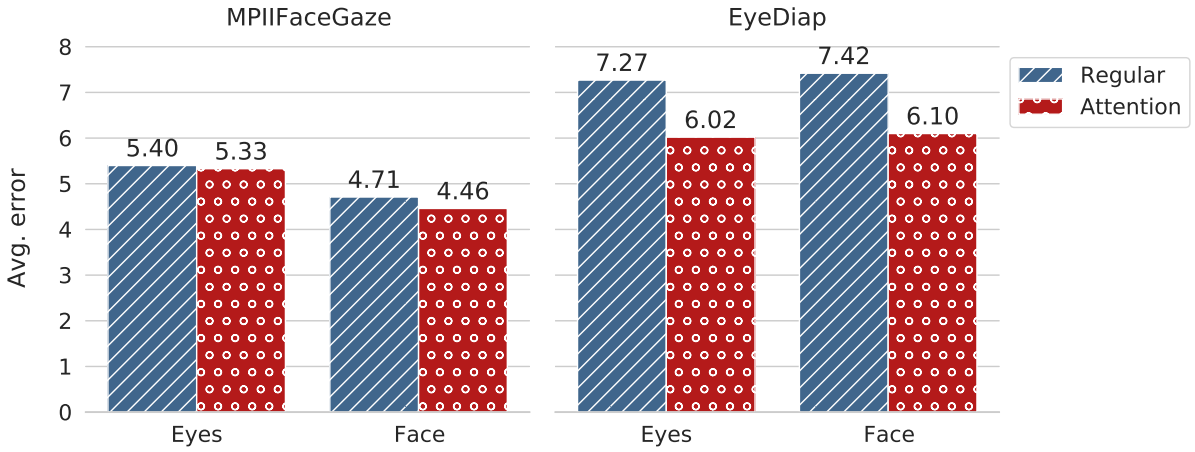


Figure 6.1: Comparison of average angular error for single-branch gaze estimation networks on the MPIIFaceGaze and EyeDiap data sets. Blue bars represent evaluations with ResNet-14 as the backbone, while red bars represent those with ARes-14.

The larger magnitude of gains on the EyeDiap data set is attributed to the fact that it is a more challenging data set with regards to head pose, which we hypothesize is an area that benefits from the use of self-attention augmentation.

The slightly larger gains on the face-only network may be explained by the fact that, while in eye images the region of interest is essentially the pupil, the regions of interest for gaze estimation in a facial picture (eyes, nose, mouth) are further apart from each other. This kind of spatial relationship is exactly where self-augmentation can be the most useful. Although the error decrease in single-branch networks seems promising, results in Table 5.3 shows that the advantage of using multi-input branches is clear.

Another analysis in Fig. 6.2 shows a comparison of the evaluation results obtained from experiments using multiple inputs across different iterations of our proposed gaze estimation architecture (replacing ResNet backbones by ARes-14 in each branch). It is worth noting that between the networks using ARes-14 as only one of the backbones, the one with self-attention augmentation on the face branch wins by a slight margin. On the MPIIFaceGaze data set, the one with attention only on the eye branch even had a small but noticeable drop in performance when compared with the regular CNN baseline. Analyzing these results in comparison with those from the single-branch networks, it is possible to note that the face branch benefits slightly more from self-attention augmented convolutions due to having more distant elements that can be correlated by self-attention. This supposition is reinforced by our results on the evaluation of mixed attention and regular convolution networks.

All in all, our findings indicate that self-attention augmented convolutions can be used as drop-in replacements to convolutional layers in gaze estimation networks to reduce the angular error in evaluation. Yet, while self-attention augmented convolutions work well with both face and eye-input images, our experiments showed that networks working with the full-face image as input were more prone to improvement when augmented by self-

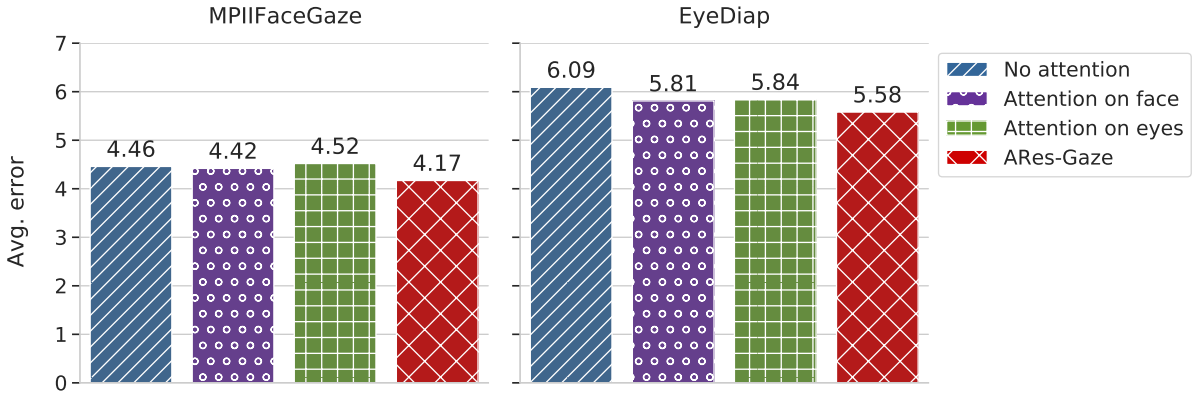


Figure 6.2: Comparison of average angular error for four different versions of the proposed gaze estimation framework. From left to right, respectively: regular convolutional baseline (blue), a version with ARes-14 on the face branch and ResNet-14 on the eye branch (purple), a version with ResNet-14 on the face branch and ARes-14 on the eye branch (green), and the fully attention-augmented ARes-gaze (red).

attention. The ARes-gaze framework which uses ARes-14 networks for both face and eye-inputs had the best results on our ablation studies, in some scenarios outperforming and at worst being comparable to state-of-the-art similar appearance-based gaze estimation methods on the MPIIFaceGaze and EyeDiap data sets.

6.2 ON THE NUMBER OF ATTENTION HEADS PER CONVOLUTIONAL LAYER

Attention-heads is the number of attention operations performed in parallel during the forward pass of an AAConv. We performed ablation studies to explore the effect of this hyperparameter on the performance of our network. The results obtained were counter-intuitive, in it that for numbers of attention-heads less than eight, $Nh < 8$, the ARes-gaze framework sometimes actually performed worse than the convolutional baseline. As detailed in Section 2.3.5, the output of a self-attention augmented convolution is the concatenation of a regular convolution and a self-attention layer, which is composed of feature maps obtained from the attention heads. We hypothesize that for lower numbers of attention-heads, the attention portion of the output may not be able to sufficiently represent relevant features, and may even harm the convolutional feature maps upon joining with them. This hypothesis is further reinforced by a visual analysis of early feature maps extracted from a trained network, as depicted in Fig. 6.3.

Figure 6.3 shows the weights computed from attention maps (one per attention head), extracted from the first augmented layer of an ARes-14 face branch during prediction. These weights are used to compute the final attention feature maps that will be joined with the convolution results on the output layer. Each row represents a different trained network with the number of attention-heads $Nh = 2, 4$ and 8, as seen in Table 5.4. It is worth observing that, for some inputs, the self-attention augmented layer is capable of highlighting semantically relevant regions of the image. We noted, however, that when

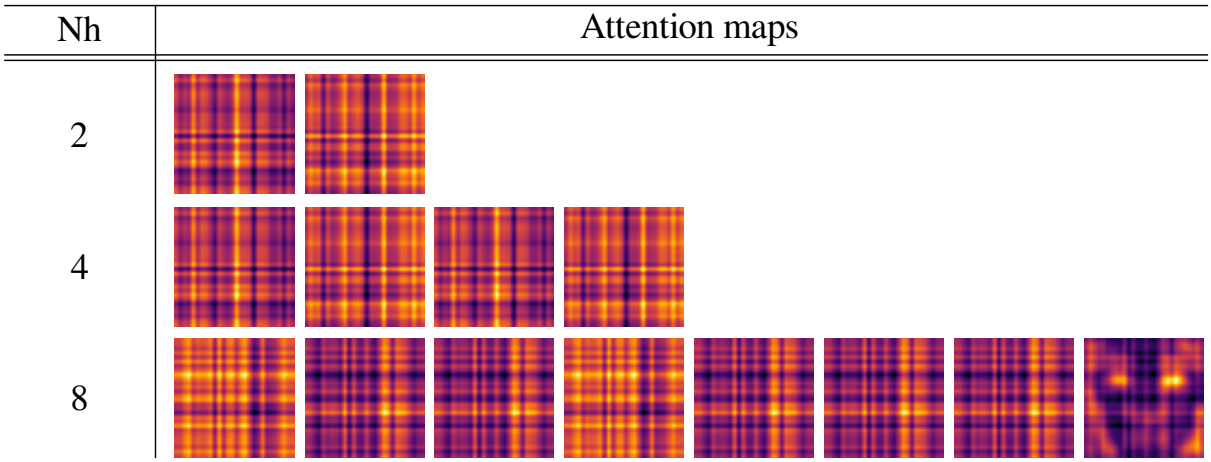


Figure 6.3: Visualization of weights for intermediate feature maps from attention heads on the first attention-augmented convolution when performing inference.

this happens, it is only on the map of the eighth attention head. This fact leads us to conclude that the attention layer might need a certain depth of attention-heads to specialize in very particular tasks. In turn, when this specialization is not reached properly, the output of the self-attention augmented convolution (concatenation of a regular convolution and an attention layer) can have lower quality than a regular convolutional layer with a larger feature space. This would explain the counter-intuitive results reached when using self-attention augmented layers with values lower than 8 for Nh, being less accurate than a regular convolutional CNN.

To more easily interpret the highlighted regions by the weights of the eight attention map, Fig. 6.4 illustrates a masked version of the input image by the map in question, after operations of thresholding and smoothing for better visualization. As expected, the eyes and eyebrows are the most relevant regions for the gaze estimation task. Notably, the nose, mouth, and background are also highlighted, which we speculate to be the source for head-pose related information implicitly used for the prediction.

6.2.1 The advantages and disadvantages of applying attention augmented convolutional layers in gaze applications

The attention-augmented approach used in the framework presented in this document was shown to improve the overall performance of gaze estimation, reducing the average angle error in all of the experiments reported. Additionally, we were able to show that the attention layers can create visually interpretable weight maps. This adds another potential use to the method that can be exploited by applications where the explainability of the task is an important factor. These features, however, come at the expense of additional computational costs.

This additional computational cost is especially problematic during the training of the networks. Without access to the resources of a highly efficient computing environment,

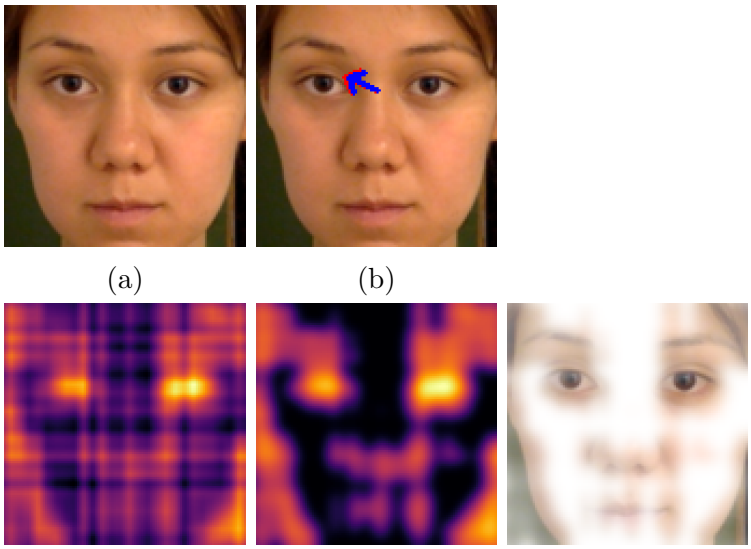


Figure 6.4: Stages of attention maps: (a) is the input image, (b) is the projected prediction (red) and ground truth (blue) vectors, (c) is the last attention map on a random pixel in the image for the first convolutional layer, (d) is the attention map after thresholding and smoothing, and (e) is an overlay of the smooth map and the input image, evidencing relevant features.

the training of the AAConv layers can be impossible due to memory consumption. This is less of a problem during the inference stage, however, which is where we will focus on in this section when we tackle the question of how can the work described here be utilized in real-work situations.

As mentioned in Chapter 1, gaze direction is used in applications spanning a wide array of systems in many different areas. For our practical analysis, these applications are separated into two categories. We split them with regards to the frequency with which they need to process data: online applications (those that need to process data constantly and produce results as fast as possible), and batch applications (those that can accumulate data to be analyzed in batches). Any system where the gaze is a means of interaction, such as mobile (BARZ *et al.*, 2018) and augmented reality (NILSSON; GUSTAFSSON; CARLEBERG, 2009) applications, for example, falls into the online category. The application needs to immediately process the input data (in this case, the user's gaze direction) to respond and produce the output that the user is expecting. Long processing times usually make users stop using these systems, rendering them next to useless.

In contrast, batch applications are those where immediate response time is not critical, and the input data can be queued to be processed in batches later. Research-related applications are usually a good example of this. For example, video data gathered from subjects performing a particular task in a controlled environment, such as looking at a screen (NAKANO *et al.*, 2010), can be stored to be processed and analyzed at any point

in the future. It is worth noting that such research applications need not be restricted to academia, and can be applied in fields such as marketing and advertisement by analyzing and predicting the gaze patterns of customers in-the-wild (RUMPF; BORONCZYK; BREUER, 2020).

During inference time, while the attention overhead of ARes-Gaze is not as prohibitive as during training, the framework still performs slower than a regular convolutional counterpart on a consumer-grade CPU. For this reason, we consider ARes-Gaze better suited for batch processing applications, where speed is not as much of a factor but it is desirable to have as much accuracy as possible. For online applications, it is acceptable to use methods that may provide less accuracy but offer better speed guarantees.

6.3 FUTURE WORK

The work presented in this document sought to investigate whether self-attention augmented convolutions could be a means to reduce the angular error in appearance-based gaze estimation. We found that the use of 2D self-attention in convolutional layers, when compared to an equivalent regular convolutional network can indeed produce more accurate results. In our experiments, we used twin ARes-14 branches as self-attention augmented CNNs in our experiments. We believe that further research is merited on the design of optimal architectures for each of the backbones for a multi-input attention-augmented framework such as the proposed ARes-gaze. We showed that face inputs had more to gain from using AAConvs than eye inputs. Incorporating domain knowledge of both attention mechanisms and gaze estimation to refine each branch for its particular input (face and eyes) may produce even better results than those reported. The spatial awareness afforded to the framework by the self-attention augmented convolutions can also be a promising way to develop joint head pose and gaze direction estimation networks. One possible way towards this goal is the possibility of including other types of input such as facial landmarks and explore their behavior in attention-augmented convolutional networks.

BIBLIOGRAPHY

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

LIN, Z. *et al.* A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

VASWANI, A. *et al.* Attention is all you need. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2017. p. 5998–6008.

ZHANG, X. *et al.* Appearance-based gaze estimation in the wild. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 4511–4520.

KRAFKA, K. *et al.* Eye tracking for everyone. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 2176–2184.

PALMERO, C. *et al.* Recurrent cnn for 3d gaze estimation using appearance and shape cues. *arXiv preprint arXiv:1805.03064*, 2018.

ZHANG, X. *et al.* It's written all over your face: Full-face appearance-based gaze estimation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. [S.l.: s.n.], 2017. p. 51–60.

FISCHER, T.; CHANG, H. J.; DEMIRIS, Y. Rt-gene: Real-time eye gaze estimation in natural environments. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2018. p. 334–352.

CHENG, Y. *et al.* Gaze estimation by exploring two-eye asymmetry. *IEEE Transactions on Image Processing*, IEEE, v. 29, p. 5259–5272, 2020.

SMITH, B. A. *et al.* Gaze locking: passive eye contact detection for human-object interaction. In: *Proceedings of the 26th annual ACM symposium on User interface software and technology*. [S.l.: s.n.], 2013. p. 271–280.

MORA, K. A. F.; MONAY, F.; ODOBEZ, J.-M. Eyediap: A database for the development and evaluation of gaze estimation algorithms from rgb and rgb-d cameras. In: *Proceedings of the Symposium on Eye Tracking Research and Applications*. [S.l.: s.n.], 2014. p. 255–258.

SUGANO, Y.; MATSUSHITA, Y.; SATO, Y. Learning-by-synthesis for appearance-based 3d gaze estimation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2014. p. 1821–1828.

HUANG, Q.; VEERARAGHAVAN, A.; SABHARWAL, A. Tabletgaze: dataset and analysis for unconstrained appearance-based gaze estimation in mobile tablets. *Machine Vision and Applications*, Springer, v. 28, n. 5-6, p. 445–461, 2017.

HU, J.; SHEN, L.; SUN, G. Squeeze-and-excitation networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2018. p. 7132–7141.

PARK, J. *et al.* Bam: Bottleneck attention module. *arXiv preprint arXiv:1807.06514*, 2018.

HE, K. *et al.* Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.

WOO, S. *et al.* Cbam: Convolutional block attention module. In: *Proceedings of the European conference on computer vision (ECCV)*. [S.l.: s.n.], 2018. p. 3–19.

HOLLANDS, M. A.; PATLA, A. E.; VICKERS, J. N. "look where you're going!": gaze behaviour associated with maintaining and changing the direction of locomotion. *Experimental brain research*, Springer, v. 143, n. 2, p. 221–230, 2002.

WARLOP, G. *et al.* Gaze behaviour during walking in young adults with developmental coordination disorder. *Human Movement Science*, Elsevier, v. 71, p. 102616, 2020.

NAKANO, T. *et al.* Atypical gaze patterns in children and adults with autism spectrum disorders dissociated from developmental changes in gaze behaviour. *Proceedings of the Royal Society B: Biological Sciences*, The Royal Society, v. 277, n. 1696, p. 2935–2943, 2010.

NILSSON, S.; GUSTAFSSON, T.; CARLEBERG, P. Hands free interaction with virtual information in a real environment: Eye gaze as an interaction tool in an augmented reality system. *PsychNology Journal*, Citeseer, v. 7, n. 2, 2009.

LANKES, M.; STIGLBAUER, B. Gazear: Mobile gaze-based interaction in the context of augmented reality games. In: SPRINGER. *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*. [S.l.], 2016. p. 397–406.

LEE, J.-Y. *et al.* Design and implementation of an augmented reality system using gaze interaction. In: IEEE. *2011 International Conference on Information Science and Applications*. [S.l.], 2011. p. 1–8.

BARZ, M. *et al.* Error-aware gaze-based interfaces for robust mobile gaze interaction. In: *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. [S.l.: s.n.], 2018. p. 1–10.

HAKKANI-TÜR, D. *et al.* Eye gaze for spoken language understanding in multi-modal conversational interactions. In: *Proceedings of the 16th International Conference on Multimodal Interaction*. [S.l.: s.n.], 2014. p. 263–266.

HANSEN, D. W.; JI, Q. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 32, n. 3, p. 478–500, 2009.

ZHU, Z.; JI, Q. Eye and gaze tracking for interactive graphic display. *Machine Vision and Applications*, Springer, v. 15, n. 3, p. 139–148, 2004.

ZHU, Z.; JI, Q.; BENNETT, K. P. Nonlinear eye gaze mapping function estimation via support vector regression. In: IEEE. *18th International Conference on Pattern Recognition (ICPR'06)*. [S.l.], 2006. v. 1, p. 1132–1135.

OHNO, T.; MUKAWA, N. A free-head, simple calibration, gaze tracking system that enables gaze-based interaction. In: *Proceedings of the 2004 symposium on Eye tracking research & applications*. [S.l.: s.n.], 2004. p. 115–122.

NOUREDDIN, B.; LAWRENCE, P. D.; MAN, C. A non-contact device for tracking gaze in a human computer interface. *Computer Vision and Image Understanding*, Elsevier, v. 98, n. 1, p. 52–82, 2005.

TALMI, K.; LIU, J. Eye and gaze tracking for visually controlled interactive stereoscopic displays. *Signal Processing: Image Communication*, Elsevier, v. 14, n. 10, p. 799–810, 1999.

TAN, K.-H.; KRIEGMAN, D. J.; AHUJA, N. Appearance-based eye gaze estimation. In: IEEE. *Sixth IEEE Workshop on Applications of Computer Vision, 2002.(WACV 2002). Proceedings.* [S.l.], 2002. p. 191–195.

WILLIAMS, O.; BLAKE, A.; CIPOLLA, R. Sparse and semi-supervised visual mapping with the sˆ 3gp. In: IEEE. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. [S.l.], 2006. v. 1, p. 230–237.

MARTINEZ, F.; CARBONE, A.; PISSALOUX, E. Gaze estimation using local features and non-linear regression. In: IEEE. *2012 19th IEEE International Conference on Image Processing*. [S.l.], 2012. p. 1961–1964.

CHEN, Z.; SHI, B. E. Geddnet: A network for gaze estimation with dilation and decomposition. *arXiv preprint arXiv:2001.09284*, 2020.

BELLO, I. *et al.* Attention augmented convolutional networks. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 3286–3295.

RUSSAKOVSKY, O. *et al.* Imagenet large scale visual recognition challenge. *International journal of computer vision*, Springer, v. 115, n. 3, p. 211–252, 2015.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105.

SZEGEDY, C. *et al.* Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 1–9.

LIPTON, Z. C.; BERKOWITZ, J.; ELKAN, C. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

CHUNG, J. *et al.* Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

ZHU, W.; DENG, H. Monocular free-head 3d gaze tracking with deep learning and geometry constraints. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2017. p. 3143–3152.

PASZKE, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2019. p. 8026–8037.

BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In: *Proceedings of COMPSTAT'2010*. [S.l.]: Springer, 2010. p. 177–186.

LOSHCHILOV, I.; HUTTER, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

GIRSHICK, R. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 1440–1448.

RUMPF, C.; BORONCZYK, F.; BREUER, C. Predicting consumer gaze hits: A simulation model of visual attention to dynamic marketing stimuli. *Journal of Business Research*, Elsevier, v. 111, p. 208–217, 2020.