University of Crete
Medical School
Bionformatics

Investigating the contributions of neural features in deep learning architectures

Panagiotis Linardos

Heraklion, September 2018

Abstract

This work investigates how certain inspirations from the brain can be either fruitful or irrelevant for the advancement of the Deep Learning field and vice versa. Firstly, inspired by the modular organization of the brain, an Artificial Neural Network is developed that learns semantic hierarchies from the ImageNet dataset. The network initially learns to recognize images of animals in contrast to random pictures; after that it is extended with newly added layers on more specific, complex classes from the same dataset (e.g., canines/felines). Furthermore, inspired by the variable activity of neurons in the brain we manipulate the learning rate of different layers to see what gives us best results. Finally, to assess catastrophic forgetting, we apply a forgetting test, assessing the performance on the previous task of simply recognizing animals. On a different level, the work explores another interesting aspect of the human brain, that is its ability to allocate attention on the salient information. This part of the work adapts a Deep Neural Network for image saliency prediction to the temporal domain of egocentric video. The saliency map is computed for each video frame, firstly using an off-the-shelf model trained on static images, then by adding either a convolutional or a ConvLSTM layer trained specifically on video saliency prediction with the large-scale DHF1K dataset. Each configuration is studied on EgoMon, a newly released dataset made of seven egocentric videos recorded by three subjects in both free-viewing and task-driven set ups.

Acknowledgements

I would like to express my deepest gratitude to all the people who helped me throughout this year in cultivating my skills and developing the project.

For the part of the thesis that was developed in Heraklion:
Research Director Panayiota Poirazi, for accepting to supervise me in her lab, allowing me the freedom to pursue the research I am interested in and providing me the necessary infrastructure to do so, supporting me with collaborations from other labs as well as providing valuable advice in the neurological aspects of the project. PhD (recently graduate) Spiros Chavlis, for supervising me and providing me with precious support throughout the progress of my thesis. His contributions on the neurological and some of the technical aspects were highly valued as well. Post-Docs Emmanouil Hourdakis and Gregory Tsagatakis for providing me with advice related to the deep learning aspects of the project.

For the part of the thesis that was developed in Barcelona:
Professor Xavier Giro-i-Nieto, for taking me in his lab through the Erasmus program and directly supervising me on a weekly basis despite his busy schedule. Post Doc and co-supervisor Eva Mohedano, for her consistent advices and contributions. Software Engineer Albert Gil, for his immense help pertaining to technical difficulties in the server.

‘Never follow your dreams. Follow your effort. Its not about what you can dream of. Thats easy. Its about whether or not its important enough to you to do the work to be ready to be successful.’

Mark Cuban

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Artificial Neural Networks (ANNs) have remarkable capabilities when it comes to extracting underlying patterns (features) from data. Stemming from neuroscientific inspiration since their inception, the evolution of these architectures still draws plenty of insights from the study of the brain [Hassabis et al., 2017, Marblestone et al., 2016]. It should be a rather obvious approach, that in the course of making an intelligent machine you should use the one example of intelligence that you already have in your hands. Following this approach you may very well end up with a better understanding of the brain itself, killing two birds with one stone. This work explores how different inspirations from the human brain may or may not support developments in ANN architectures.

One aspect that was explored in this work was the creation of a network that will be able recognize a set of classes that is hierarchically structured. Consider that, when it comes to separate classes, there are always common low level-features such as edges. In the brain there is a distinction between the active neurons when observing such simple features and those that are active upon more complex patterns which are called simple and complex cells respectively [H. and N., 1962, Essen et al., 1992]. Commonly, an adult does not relearn to recognize edges every time a new class is observed, so why propagate the error all the way to the start in an

ANN structure every time that is trained on a new class? Furthermore, as complex cells are activated selectively by particular classes they form synapses with other complex cells that fire when observing these classes. In the end subnetworks are formed and encode distinct classes. One consideration to keep in mind is that when the boundaries between distinct tasks grow more and more abstract even these subnetworks tend to communicate.

Setting aside the biological inspiration, modularity can be thought of as a case of divide and conquer. A complex computational task can be subdivided into simpler subtasks which are accomplished by a number of specialized local computational systems. Instead of one monolithic network learning one complex mapping you end up with individual modules that learn a, possibly simpler, set of mappings. Also, as was previously mentioned, the homogeneous connectivity in monolithic neural networks may result in a lack of stability of representation and is susceptible to interference which may result in catastrophic forgetting. By introducing modularity you avoid such interference resulting in a more robust and fault tolerant neural network model. [Farooq, 2000]

A different aspect that was explored during the course of this thesis was the task of saliency prediction. The term saliency pertains to how an object or any piece of information may stand out from its surroundings. Detecting saliency is an integral part to how sentient organisms process information. We live in a world where the data we receive on everyday basis is immense and cluttered with noise. The brain has, therefore, developed a mechanism that allows living organisms to allocate their attention on the most relevant information, so as to function efficiently. This particular study is focused on prediction of saliency pertaining to visual information; however, saliency detection is relevant to all kinds of modalities. More specifically, this project extends an existing model to the task of video predictions. As the brain is accustomed to perceive time, we hypothesize that videos provide a better testbed for bio-inspired studies.

1.2 Related Work

1.2.1 Brain-inspired Deep Learning Methods

Perhaps the most iconic architecture that has been hugely influenced by brain mechanisms and is now widely used on many applications are Convolutional Neural Networks (CNN). They became popularized after their huge success on 2012 where a CNN architecture named AlexNet [Krizhevsky et al., 2012] outperformed conventional methods by a margin of more than 10% on the ILSVRC (ImageNet Large-Scale Visual Recognition Challenge). However, the roots of this architecture stem from further back in the past; the pioneering work that first introduced them was done by [LeCun et al., 1998a]. For the development of this architecture the authors were hugely inspired by the visual system of the brain, where simple cells detect local features and complex cells pool the outputs of simple cells within a retinal mapping [Hubel and Wiesel, 1962]. Computer vision research is hugely reliant on these networks now, and indeed it is really hard to find a recent architecture that does not utilize CNNs to some extend.

Insights from biology have also resulted in the creation of an interesting regularization scheme, which allows networks to generalize more efficiently beyond training data. The scheme is called "dropout" and the way it works is, for each training example, to turn off neurons randomly with a specified probability [Hinton et al., 2012]. As only a subset of units participate in the processing of a given training example, co-adaptation of the weights is reduced and they end up more independent of each other. This approach was motivated by the stochasticity that is inherent in biological systems, as neurons in the brain also have a chance of not firing immediately after they have fired from receiving inputs.

Lastly, we will mention one more biological inspiration that is hugely applied and has allowed ANNs to be extended to even deeper architectures. This inspiration pertains to the choice of activation function. Common activation functions such as sigmoid and hyperbolic tangent include plateau segments, where they saturate. Neuroscience literature, however, has illustrated that this is rarely the case with cortical neurons; thus, an activation function approximated by

a rectifier was deemed more appropriate. Applying this type of non-linearity to neural nets in the form of a "rectified linear unit" (ReLU) produced positive results and enabled training of much deeper nets [Glorot et al., 2011].

1.2.2 Video Saliency Prediction

The success of Deep Neural Networks for solving computer vision tasks has been recently explored in the context of video saliency prediction. These works have basically applied to this domain the architectures developed for video action recognition. Past work includes models based on complex design choices such as the two-stream network architecture [Bak et al., 2017] which segregates the learning of spatial and motion information, or the two-layer Long Short-Term Memory Network (LSTM) with object information [Simonyan and Zisserman, 2014a]. A shortcoming in these methods is that they do not utilize the existing large-scale static fixation datasets.

A more recent approach, however, has been shown to produce better results with an arguably simpler architecture based on the integration of an existing state-of-the-art saliency model in conjunction with a ConvLSTM module [Wang et al., 2018]. An interesting novelty in that work was the incorporation of a supervised attention mechanism into the network structure. Meanwhile, a VGG16 architecture extracts a deeper representation of the input frame (a VGG architecture refers to a commonly used convolutional architecture [Simonyan and Zisserman, 2014b] and the number 16 implies that it uses 16 layers). The intra-frame salient features extracted by the supervised attention mechanism are then enhanced on this representation frame by element-wise multiplication. The final output is fed to a sequential model for detection of the inter-frame temporal features.

Another recent work [Gorji and Clark, 2018] has achieved state-of-the-art in video saliency prediction using a ConvLSTM structure with 4 path ways. The structure is focused to work on videos that include other actors. Specifically, there is a pathway that follows the gaze of other actors in the scene and two pathways for when there is a rapid change, which can either be an actor leaving the scene or an entire scene change. These pathways hold no particular interest in

the study of egocentric vision as there are no real rapid scene changes in real life and there is not necessarily always a focus on other people (the kitchen setting is a good example where your attention is entirely focused on objects). Their work however also uses one other pathway called the "saliency pathway" which is simply a state-of-the-art saliency model in conjunction with a ConvLSTM. In their "saliency pathway" the output of the static saliency predictor is directly fed to the ConvLSTM rather than using it as an enhancer to a deeper frame representation.

In line with this reasoning we focused on another state-of-the-art model, that has not been tested in these works and has been shown to outperform most of the used saliency predictors, namely SalGAN [Pan et al., 2017]. We used a simple approach, by utilizing the output of this network as input for the 2 extensions of the model: One simple convolutional layer and one ConvLSTM, which should be able to extract temporal patterns as well. Furthermore we evaluated our model on egocentric vision; using the newly released EgoMon dataset [Linardos et al., 2018].

Regarding egocentric saliency prediction with deep models, the most relevant work had been done recently by [Huang et al., 2018]. In their work, Huang and colleagues propose to model the bottom-up and top-down attention mechanisms on the GTEA Gaze dataset (http://www.cbi.gatech.edu
Their approach is much more complex, as it combines a saliency prediction with a task-dependent attention module, and focuses entirely on a cooking dataset. Here we hope to make some interesting comparisons to a variety of tasks that correspond to both task-driven and free-viewing settings.

Chapter 2

Background Theory

2.1 Deep Neural Networks

2.1.1 Feedforward Neural Networks

The most basic form of a deep neural net is what we call a feedforward neural network. In its essence the network learns to approximate a function $\boldsymbol{f}$. This function is mapping an input to an output; in the case of classification it maps an input x to an output category y. In truth, this function is a chain of functions, where we can imagine that each function component is another layer of the network. Each of these function components is a linear transformation followed by a non-linear activation function. Breaking these down to equations:

A linear transformation with learnable parameters w (weight) and b (bias) can be defined as:

$$z(x) = x^T w + b \tag{2.1}$$

This equation is then followed by an activation function *a(z(x))*, which introduces the non-linearity. These two components form a layer of the neural network. We can denote a layer as g^{n} where n denotes the depth of the layer. In a three-layered neural network, for example, the function $\boldsymbol{f}$ to be approximated would be equal to $g^3(g^2(g^1(\mathrm{x})))$

So, essentially, we introduced how information flows forward in a neural network. For a network to train and learn the parameters to best approximate the function, however, information needs to flow backwards as well. What we mean by "learning" is how the network changes these parameters in order to minimize a certain cost function J. For the cost of a single training example, you take the predicted output of the network along with the ground truth and compute their distance. Computing the cost for all training examples and then averaging the results produces the total cost of the network. The negative gradient of this cost function with respect to each individual parameter tells us what is the most optimal way to update the parameters in order to decrease the cost.

Figure 2.1 illustrates how the optimization would look if we only had 1 parameter to tune, picturing the cost function as a ball sliding down a slope. In this simple example it's easy to think how the weight should decrease if the derivative dJ/dw is positive, and increase when it's negative. One may get carried away into thinking that it would be simple to just calculate the position where dJ/dw becomes 0 and be done with it; however, in many dimensions this is not feasible. It should be noted, therefore, that since we have a multitude of parameters in a neural network we need to appreciate that gradient descent works in a multi-dimensional field. Naturally, it is impossible to think in more than 3 dimensions, but multi-variate calculus tells us that the gradient of a function in such a space gives us the direction of steepest ascent. In other words by calculating the gradient we know in which direction the function would increase most quickly. Since in our case we want to minimize the cost function, we take the negative of the gradient.

The method that made it possible for neural networks to train on a pragmatic amount of time is backpropagation [Rumelhart et al., 1986]. Back-propagation is, basically, an algorithm that computes the derivatives of a chain function, with a specific order of operations that is highly efficient. There is commonly misconceptions pertaining to how gradient descent and backpropagation relate to each other. To clarify, the term backpropagation is used to describe the evaluation of derivatives. The goal of backpropagation is to compute the partial derivatives $\partial J/\partial w$ and $\partial J/\partial b$ of the cost function J with respect to any weight w or bias b in the network. These derivatives are then used by gradient descent to readjust the weights.

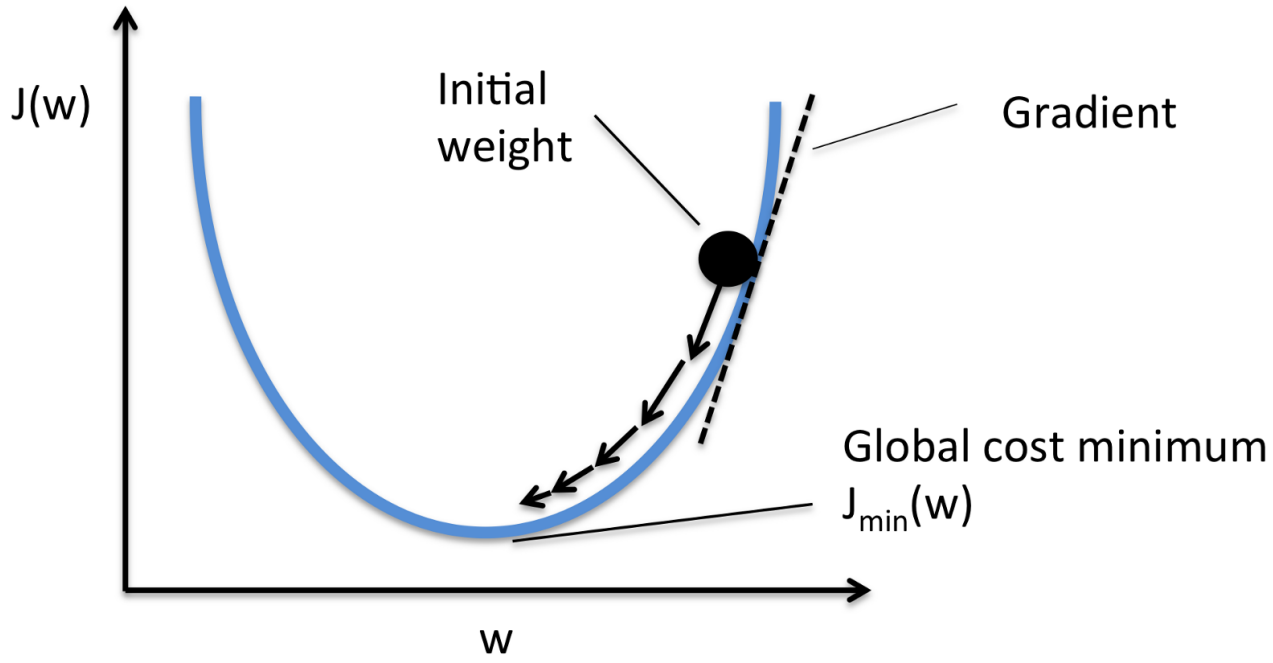


Figure 2.1: Gradient Descent schematic.[https://hackernoon.com/gradient-descent-aynk-7cbe95a778da]

We will be using the binary cross entropy loss function in this project (equation 2.2); however, to further explain backpropagation equations, we will borrow the mean squared loss function (equation 2.3), as it is arguably the easiest one to demonstrate the equations with.

$$J(x) = -\frac{1}{N}\sum_{i=1}^{N} y_i log(a^{(\mathrm{L})}(x_\mathrm{i})) + (1-y_i) log(1-a^{(\mathrm{L})}(x_\mathrm{i})) \tag{2.2}$$

$$J(x) = -\frac{1}{2N}\sum_{i=1}^{N} \|y_i - a^{(\mathrm{L})}\|^2 \tag{2.3}$$

Here L represents the total number of layers and $\mathrm{a}^{(\mathrm{L})}$ corresponds to the activation of the final layer L. Note that the activation function cannot be changed, but we may tune the weights and biases upon which the activation function's value relies on. To compute, for example, the partial derivative of the cost function with respect to the weight on layer L, we will be using the chain rule. Note that from this point on, we refer to a single training example and so the N term becomes equal to 1.

$$\frac{\partial J}{\partial w^{(\mathrm{L})}} = \frac{\partial J}{\partial a^{(\mathrm{L})}} \frac{\partial a^{(\mathrm{L})}}{\partial z^{(\mathrm{L})}} \frac{\partial z^{(\mathrm{L})}}{\partial w^{(\mathrm{L})}} \tag{2.4}$$

Given that our cost function is the mean squared error and our activation is a sigmoid, the derivatives will boil down to:

$$\frac{\partial J}{\partial a^{(\mathrm{L})}} = (a^{(\mathrm{L})} - y) \tag{2.5}$$

$$\frac{\partial a^{(\mathrm{L})}}{\partial z^{(\mathrm{L})}} = a^{(\mathrm{L})}(1 - a^{(\mathrm{L})}) \tag{2.6}$$

$$\frac{\partial z^{(\mathrm{L})}}{\partial w^{(\mathrm{L})}} = a^{(\mathrm{L\text{-}1})} \tag{2.7}$$

Similarly the partial derivative to bias can be computed. The partial derivatives to parameters of shallower layers may be computed as well, simply by iterating the chain rule idea backwards further. This way we are calculating the sensitivity of the cost function to changes pertaining to all parameters of the network, so that the update step may nudge all of these parameters to towards the most optimal direction.

2.1.2 Convolutional Neural Networks

As it was mentioned in the Introduction, Convolutional Neural Networks (CNNs) have seen huge success in practical applications, particularly in relation to computer vision. What makes them stand out is that they deploy a convolution operation instead of matrix multiplication, which gives them an advantage in processing data that has a known, grid-like topology (such is the case with images, which can be thought of as 2-D grids of pixels).

A convolution operation is essentially an integral that expresses the amount of overlap of one function g when it is shifted over another function f.

$$(f * g)(x) = \int f(t)g(x - t)dt \tag{2.8}$$

We may now describe a picture of how convolution works on matrices. Given matrices F, G (with G being of smaller size than F), G shifts over F and element-wise matrix multiplication

is performed between the overlapping elements. From this matrix multiplication a value is produced and then G is shifted again. The aggregation of values from these operations compose the output matrix of the convolution.

In the case of convolutional neural networks, G is called a kernel (or a filter) and its values are the learnable parameters of the network. When we describe a convolutional layer we mainly define 3 things: the amount of filters it contains, the size of these filters and their stride (the size of their step on one shift). The number of filters defines the number of representations this layer extracts from its input. For example, if we feed a grayscale image to a convolutional layer with 128 filters then this results in 128 channels (128 representations of the same image). The case of the grayscale image, where we have only one channel, is simple; however, if we had fed an RGB image instead, there are 3 channels (red, green, blue) which means 3 matrices of pixel values to process. In this case the layer's kernels would have a depth of 3 to process all the information (in this case the depth of 3 means the same kernel will be applied 3 times, one in each channel). Similarly, the layer that follows after the one with 128 channels will use kernels of depth 128.

Overall, what makes convolutions really efficient is parameter sharing and sparsity of connections. Parameter sharing pertains to how the kernel with the same parameter processes all parts of the images as well as different depths of the image. We can understand why this would be useful, if we consider how edges repeat on different segment of the image. Even complex items, such as eyes, can appear twice in different orientations. Sparsity on the other hand, refers to how each value of an output map is influenced only by a local segment of the input. In contrast to fully connected layers, these two characteristics hugely alleviate the parameter load of the model.

A typical layer of a CNN consists of mainly three things: the convolution operation, a non-linear activation (usually the ReLU [Glorot et al., 2011]) and finally a pooling function. A pooling function is used to downsample the feature map extracted by the convolution. This function is fixed; it has no learnable parameters. Using a window of predefined size, it computes one value which may be the maximum or the average value of the elements overlapping with this window.

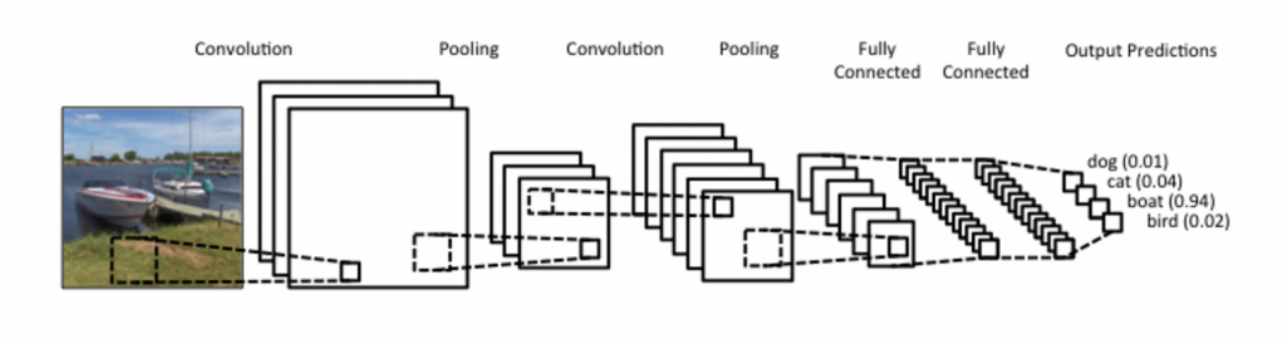


Figure 2.2: CNN schematic. Notice how the representations are down-scaled after pooling operations and how their depth changes after convolutions. [https://www.kdnuggets.com/2015/11/understanding-convolutional-neural-networks-nlp.html]

The pooling operation has been criticized for discarding valuable information; nevertheless, its contribution is mandatory, as it significantly increases the receptive field of deeper layers.

A complete CNN architecture is illustrated in figure 2.2. Essentially, the convolutional part acts as a feature extractor and the fully connected layers combine these features to make a prediction. Keep in mind that the majority of computations occur in the fully connected layer due to their high density in parameters. Depending on the task the fully connected layers may not be a necessary component. For example, when we wish for the network to output another image, we have to use another convolution operation in the last layer, instead.

2.1.3 LSTM Neural Networks

Similarly to how a convolutional architecture specializes in processing a grid of values, recurrent neural nets (RNNs) are experts in dealing with sequential data. Sequential data may be relating to speech, text, music or even DNA sequence analysis to name just a few; however, in this work we only use it in the processing of videos. To understand the intuition behind recurrent architectures, it suffices to think how humans process sequential data, like text. Similarly to how a human will continue reading the text without throwing away information from previous words, a recurrent network includes a loop that allows information to persist. We can think of recurrent nets as a simple network that also has the particularity of feeding its hidden representations to a copy of the same network.

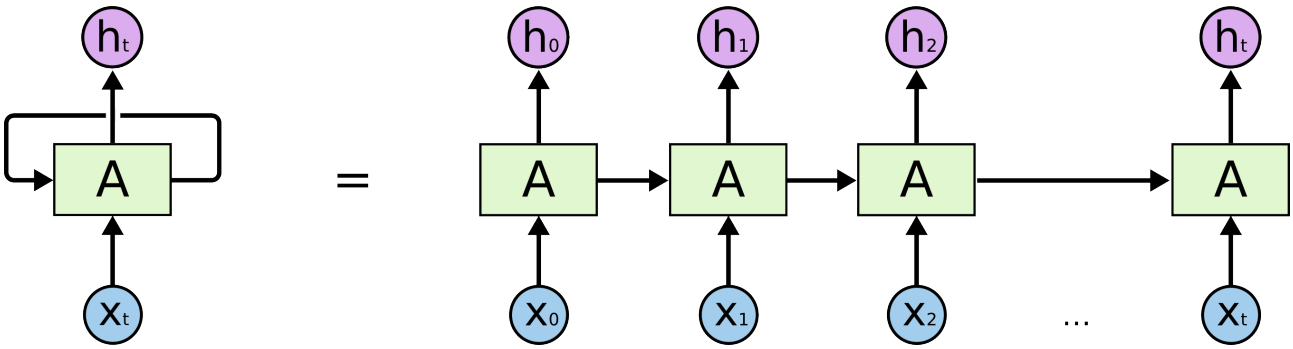


Figure 2.3: An unrolled recurrent neural network. "A" represents a typical network, but as we can see there is the particularity of feeding subsequent hidden layers with its output as well. [https://colah.github.io/posts/2015-08-Understanding-LSTMs]

Despite their success in many tasks, RNNs still faced issues when it came to storing long term relations. This motivated the creation of gated units that allow for selective forgetting of information, with parameters that are also learned on train time. In the particular architecture that has been used in this project, namely LSTM [Hochreiter and Schmidhuber, 1997], there are 2 types of information passed through from one cell to another: the hidden state and the cell state. The cell state can be allowed to pass from a network cell to another with nothing changed, enabling the learning of long term dependencies. Every cell has 3 gates, with whom it may manipulate the cell state. The forget gate is fed a concatenation of the input and the hidden state of the previous layer and outputs a matrix with values between 0 and 1 (using a sigmoid function), representing what to keep and what to throw away (equation 2.9). In the update gate, a similar matrix is learned, which decides which candidate values C to update the cell state with (equations 2.10 and 2.11). The final gate is called "output" gate and it does not affect the cell state directly, but uses a filtered version of the cell state (equation 2.12) to output the hidden state (equation 2.13).

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.9}$$

$$u_t = \sigma(W_u[h_{t-1}, x_t] + b_u) \tag{2.10}$$

$$C_t = f_t \circ C_{t-1} + u_t \circ tanh(W_C[h_{t-1}, x_t] + b_C) \tag{2.11}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.12}$$

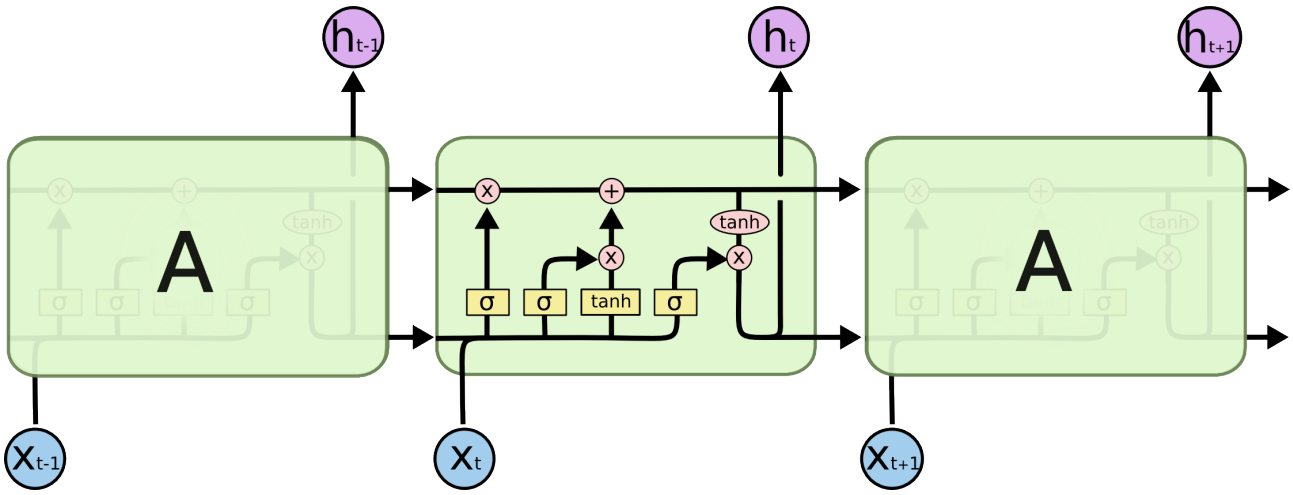


Figure 2.4: LSTM network schematic. [https://colah.github.io/posts/2015-08-Understanding-LSTMs]

$$H_t = o_t \circ tanh(C_t) \tag{2.13}$$

LSTM networks have been proven to learn long-term dependencies and outperform recurrent architectures and achieve state of the art performance on many challenging settings such as machine translation [Sutskever et al., 2014] or speech recognition [Graves et al., 2013].

2.2 Saliency

Selective visual attention has allowed organisms to direct their gaze on significant information in their environment (like an approaching predator or a vulnerable prey). The neurological basis of saliency prediction in living organisms has deployed a framework of two main components: task-driven or top-down saliency, which is governed by higher regions of the brain as the viewer is mentally engaged in what he is attending [Itti and Koch, 2001, Buschman and Miller, 2007] and bottom-up saliency, which occurs heavily when the mind wanders and so attention is driven by cues in the environment. In the end, both mechanisms work in parallel but as top-down requires voluntary effort, it requires more time and so bottom-up saliency tends to act first (50ms compared to around 200ms for top-down) [Itti and Koch, 2001].

Furthermore, evidence suggests that selective attention is coded explicitly in the cortex and separately from the visual features, which strongly indicates the existence of an explicit saliency

map in the brain [Ptak, 2012, Itti and Koch, 2001]. We are still far from a complete understanding of how the brain chooses to allocate attention, but current architectures for predicting saliency maps may help us in utilizing this aspect of the brain as a soft-attention guide for other tasks.

More specifically, models that learn to predict saliency may estimate which regions of an image have a higher probability of being fixated by an observer. The result of such predictions is expressed under the form of a heat map in which higher values are aligned with more salient pixel locations. This information can be utilized in many applications, including higher quality coding of the salient regions [Zhu and Xu, 2018], spatial-aware feature weighting [Reyes et al., 2016], or image retargeting [Theis et al., 2018]. Predicting saliency has been studied mainly in set ups where the viewer is asked to view an image [Itti et al., 1998, Harel et al., 2006, Judd et al., 2009, Borji, 2012] or video [Wang et al., 2018] depicting a scene. In these works, however, the focus is entirely on the bottom up saliency. A more complete model of attentional control should be able to effectively predict saliency that arises from top-down mechanisms. Taking advantage of such mechanisms, may prove to be fruitful for allocating attention of the network on a particular task, like a human would.

2.3 Egocentric Vision

In most cases, datasets are collected by a seated viewer observing an image or a video. Luckily, a more interesting type of dataset has emerged in the recent years, where the viewer is actively engaged with the scenery [Damen et al., 2018, Su and Grauman, 2016, Fathi et al., 2012, Linardos et al., 2018]. In this case, the viewer is wearing a wearable eye tracker that records whatever the viewer is watching along with his gaze fixations. The user is not only free to fixate the gaze over any region, but also to change the framing of the scene with his head motion. This type of dataset is particularly interesting as in many tasks such as cooking the user is mentally engaged with the scene, resulting in a higher prevalence of top-down saliency. When collecting datasets, this set up also differs from others in which the same image or video is

shown to many viewers, as in this case each recording and scene is unique for each user.

This comes in hand with certain technical difficulties, as instead of a heat map we end up with a single pixel gaze location per frame of the recording. Indeed training a model on one pixel location is particularly hard; and at this time, datasets that combine large scale and diversity do not exist in this setting.

Chapter 3

Methodology

3.1 Building a Modular Network

Based on the insights that were mentioned in Chapter 1, we implemented a neural network building on top of an existing state-of-the-art deep CNN architecture, namely ResNet [He et al., 2016]. This particular architecture consists of certain blocks, that do 3x3 convolution operations interspersed by a skip connection (Fig. 3.1). Formally, this building block can be defined as:

$$y = F(x) + x \tag{3.1}$$

Where *F(x)* is a chain function of 2 layers interspersed by a ReLU activation function, where each layer consists of a 3x3 convolution operation, followed by a batch normalization.

We decided that this architecture is more appropriate for our case compared to other state-of-the-art networks due to two reasons:

- First, the skip connections are in coherence with what happens in the brain. Indeed, it is known that neurons from lower levels of the brain wire not only to their immediate neighbors but form connections with higher level cells as well.

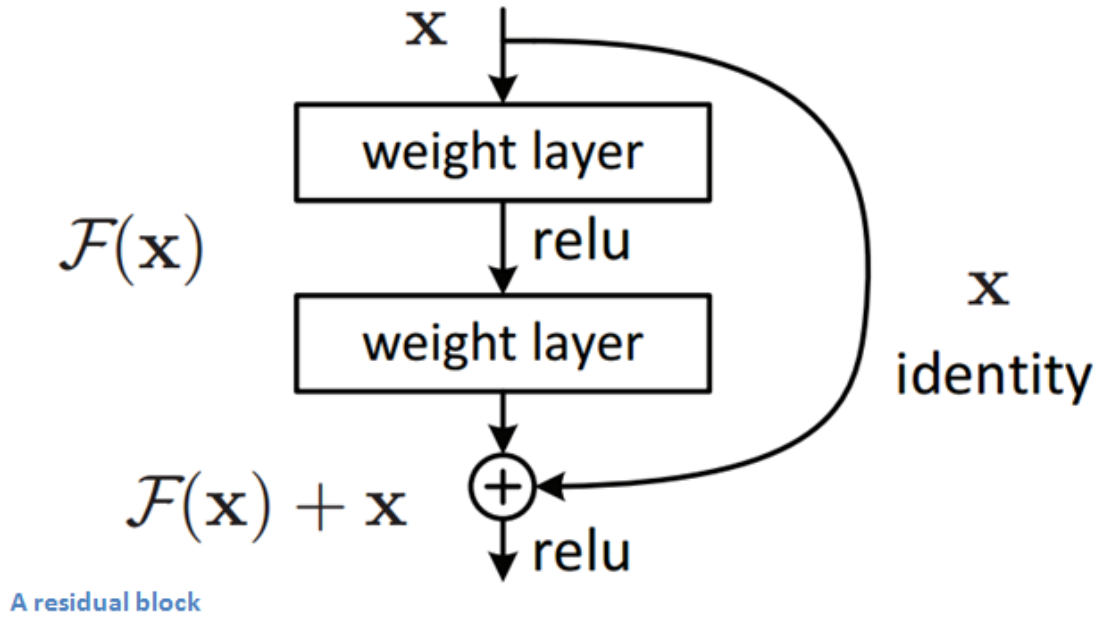


Figure 3.1: A basic residual block.

- Second, our architecture's plan is to keep growing as more complex classes are introduced, with more modules added each time. The skip connections allow the network to learn the identity function, which supports the building of deeper networks.

Our first module consists of 3 basic blocks, of the same size of layers as in the original ResNet model. The dataset used for training is ImageNet [Deng et al., 2009] a large visual database with integrated hierarchical structure in the organization of the data (http://image-net.org/). Particularly, in ImageNet it is easy to download the Animal category separately, then the Canine category and then the Feline. The categories mentioned were used for our experiment on modular learning. The first module was trained on the task of recognizing animals from a subset of ImageNet that consists of Animals (felines/canines) and random pictures (flora, rock, geological formations, fungi, construction sites). To avoid bias towards an overrepresented class we balanced the sample sizes to be equal.

After training on that task, we moved on to the task of discriminating between the more complex "canine" vs "feline" classes. We used 3 different strategies: freezing the shallow layers, varying the learning rates of different layers, and conventionally training the full extended architecture. Note that varying the learning rates was inspired by how different levels of the brain have

different levels of activity. In each task, data was sampled randomly from the bigger class so that it has an equal number of examples as the smaller one, thus avoiding bias towards the overrepresented class. Regarding hyper-parameters, we dynamically decreased learning rate through training using a decay rate of 0.1 and a starting learning rate of 0.1 (in the baseline). We loaded data using batches of size 256 and the model was trained for a total of 72 epochs. Finally 0.0001 weight decay was applied and 0.9 momentum. The model has been made publicly available on github: [https://github.com/Linardos/ModNet].

As was mentioned in the introduction, we hoped to avoid catastrophic forgetting by utilizing modular networks. Catastrophic forgetting has been shown to be a significant problem when it comes to multi-task or lifelong learning. In their work [Goodfellow et al., 2013] used dropout as an effective regularizer against catastrophic forgetting. Here we hoped that the modularity itself and the various strategies for learning may tackle this issue.

The metrics that were used were accuracy, sensitivity and specificity. These metrics are ideal on tasks of binary classification:

- **Accuracy** measures the percentage of successful predictions of the model compared to the entire data size. It is a summation of the true positives (TP) and true negatives (TN) in the output, divided by the total sample size. A downside of accuracy is that it is very sensitive to the prevalence of one class over another; however, since we are balancing the sample sizes this was not a problem.

- **Sensitivity** calculates the number of true positives divided by the entirety of positives in the data. This metric will tell us what percentage of animals were correctly predicted as animals and what percentage of canines were correctly predicted as Canines.

- **Specificity** calculates the number of true negatives divided by the entirety of negatives in the data. This metric will tell us what percentage of random pictures were correctly predicted as random and what percentage of Felines were correctly predicted as felines.

3.2 Building a Temporally-aware Saliency Predicting Network

3.2.1 Datasets

For the training of our extension we used a new large-scale video dataset for saliency prediction called DHF1K [Wang et al., 2018] . DHF1K contains 1000 videos of variable length (from 400 frames to 1200 frames at 30fps), which capture representative instances, diverse contents and various motions, with human eye-tracking annotations. From the 1000 videos, 700 are annotated and the other 300 are held back by the authors for benchmarking purposes. Note that we used a state-of-the-art network that has been trained on another dataset named SALICON [Jiang et al., 2015]. This is simply a large scale image dataset for saliency.

Furthermore, we evaluated on the newly released EgoMon dataset [Linardos et al., 2018], which pertains to egocentric vision. This dataset is composed of only 7 videos of 30 minutes duration. Egocentric datasets are still in their infancy so that one will hardly find a larger scale dataset. What makes this particular one appealing though, is the high diversity in tasks, which include settings that mentally engage the viewer (cooking, attending a presentation, playing cards) and free-viewing settings (walk in the park, strolling in the botanical gardens, a bus ride, walking to an institute). This allowed us to evaluate the model's performance on both of the fundamental components of saliency prediction.

3.2.2 Video Preprocessing

To process a video dataset, we first need to extract the keyframes. After this process the dataset may be loaded in the model the same way as it would load a batch of images. Every video will constitute a folder of frames and because we wish to learn temporal patterns, the frames have to be loaded sequentially to the model.

To do this we used a popular python computer vision library called opencv. We extracted a number of frames that correspond to the FPS of the video. For the DHF1K dataset the fps is 30, so that means 30 frames per second of the video. In the case of EgoMon this number is 15.

3.2.3 Extending a saliency predicting network

Our proposed architecture starts by processing each video frame separately with SalGAN [Pan et al., 2017], an image-based saliency prediction network that has been pre-trained on the SALICON dataset [Huang et al., 2015]. SalGAN is based on the generative adversarial network architecture, which was first introduced by [Goodfellow et al., 2014]. In this scheme, two networks are trained in conjunction: one is generating an output and the other one is trying to discriminate whether the generated output is real or not. The generator in the case of SalGAN is trained to create a convincing saliency map. To further stabilize training, the binary cross entropy loss has been used in conjunction with the adversarial loss. SalGAN has reached state-of-the-art performance in saliency. Despite the seemingly complex architecture during training, what is highly appealing in SalGAN is that during inference we no longer require the discriminator. For our goals we used the pre-trained generator; more specifically, we simply used it for inference of saliency maps from videos for further processing by our extension. This was also the first time the model has been tested on videos.

The pre-trained SalGAN generator was given an input video $\{I_t\}_t$, and produced a sequence of saliency maps $\{S_t\}_t$. Then the maps were fed into a ConvLSTM [Shi et al., 2015] as input. An LSTM was already described as an autoregressive architecture that controls the flow of information in the network using 3 gates : update, forget, output. That way, at time step t, the network may control which information from the saliency maps is relevant to keep or discard (refer to equations in chapter 2.1.3). In the case of ConvLSTMs, the difference is that the matrix multiplication operations on the input and hidden state at each gate are replaced by convolutions. To obtain a saliency map a 1x1 convolution was used at the output hidden state, so as to filter out all channels but one and end up with

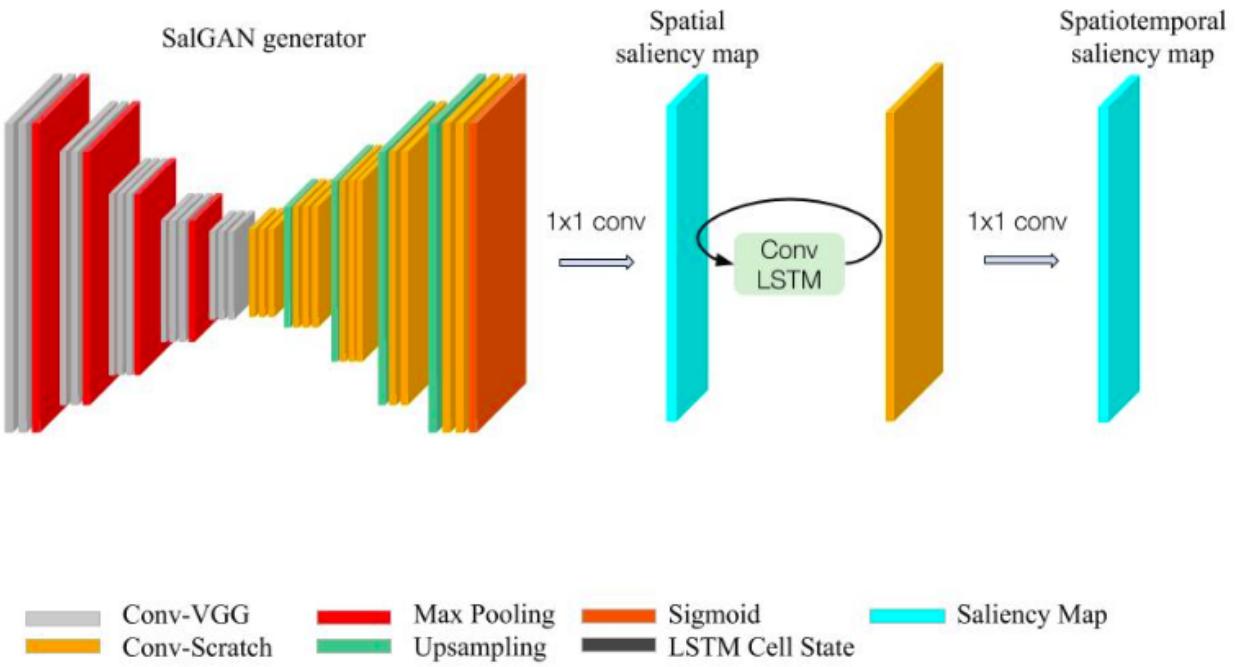


Figure 3.2: The architecture of the SalGAN extension as a temporally-aware model is presented here. Note that this is the ConvLSTM extension; the static model uses plain convolutions without the LSTM temporal recurrence.

a grayscale image (a heat map). We sequentially pass saliency maps to the ConvLSTM input and, hence, obtain a sequence of time-correlated saliency maps. Our ConvLSTM model then learned to leverage these temporal features during training. SalGAN outputs a sequence of static saliency maps which were fed into two types of adaptation layers: 128 convolutional filters [LeCun et al., 1998b] of kernel size 3x3 and padding of 1, and its temporal-aware counterpart as ConvLSTM [Shi et al., 2015] with the same convolutional parameters. Their parameters were estimated from 700 training videos from the DHF1K dataset [Wang et al., 2018]. An SGD optimizer with 0.9 momentum was used, and the learning rate started at 0.00001 and decayed with a 0.1. There was also a weight decay of 0.0001.

We also used another extension. In this case, we omitted the LSTM characteristic of the network and simply extended with a convolutional layer of the same channel dimensions (128). This can also be thought of as an ablation study, where we see if the memory network is truly relevant in this case.

We wish to study how our model performs on the domain of egocentric vision as well,

which presents the particularity of having the viewer immersed in the scene. This set up also differs from others in which the same image or video is shown to many viewers, as in this case each recording and scene is unique for each user. This results in a very challenging annotation, where each frame is annotated by a single pixel gaze location rather than a heat map. Egocentric saliency prediction has been studied in the past [Fathi et al., 2012, Su and Grauman, 2016], and here we extended this research line by assessing the state-of-the-art model in image saliency prediction to this egocentric video set up. We also assessed our extended versions of that model.

A new large-scale egocentric dataset named "EPIC-KITCHENS" was introduced in this year's ECCV conference [Damen et al., 2018]. This particular dataset focuses entirely on the task of cooking making it a very interesting top-down saliency testbed. The authors have not made the labels public yet as the dataset is part of 3 challenges: action recognition, action anticipation and object detection. We extracted the saliency maps for the EPIC-KITCHENS and made them publicly available for the community to use in this tasks [https://imatge-upc.github.io/saliency-2018-videosalgan/]. We believe that this contribution will spark a higher interest in using saliency as a soft-guide for various tasks. In the website we also include saliency maps for DHF1K and EgoMon; the trained models are may be found in our repository: [https://github.com/imatge-upc/saliency-2018-videosalgan].

3.2.4 Metrics

Evaluating a saliency model's prediction is still an ongoing research problem [Bylinskii et al., 2018]. For this reason, scientists are commonly using a variety of metrics to assess their saliency model's performance. In this work, we chose to work with the following metrics: Area under ROC Curve - Judd (AUC-Judd), Shuffled AUC (sAUC), Normalized Scanpath Saliency (NSS), Similarity (SIM), Pearsons Correlation Coefficient (CC). The reason we chose these particular metrics is because we wanted to benchmark our work against [Wang et al., 2018], where state-of-the-art saliency models have been trained on DHF1K and evaluated on these same metrics.

Table 3.1: Saliency metrics.

	Location-based	**Distribution-based**
Metrics	AUC-J, sAUC, NSS	SIM, CC

During evaluation, the calculation of these metrics has proven to be quite time-consuming. Note that we are evaluating performance from saliency maps produced by 3 different models: The vanilla SalGAN, the ConvLSTM and the Convolutional layer. To bypass this issue, we implemented a simple parallel programming scheme with 8 threads, so that 8 predictions may be evaluated at a time. This alleviated the time barrier and allowed for the evaluation of a model in approximately 2 days duration.

Saliency metrics are divided in two main categories: those that consider ground truth images at discrete fixation locations (location-based metrics) and those that consider them as a continuous distribution (distribution-based metrics). The metrics used in this work are summarized in table 3.1. This distinction makes it impossible for distribution-based metrics to assess the performance using discrete fixation maps as ground truth. For the evaluation of EgoMon, for example, it is impossible to use the similarity (SIM) metric or the correlation coefficient (CC). The reason for this, is that in egocentric vision datasets the ground truth for each frame is a single gaze fixation; therefore, it cannot be considered a continuous distribution. In the end, for the evaluation of EgoMon we used only the NSS metric, as we didn't manage to calculate the AUC and sAUC due to technical difficulties.

Below we provide a brief description for each of the metrics used here:

- Area under ROC Curve - Judd (AUC-J): Given various threshold levels it calculates performance of the saliency model as a binary classifier; where each pixel of the saliency map is considered the classification prediction. The ROC curve is created by measuring the true and false positive rates under different thresholds.
- Shuffled AUC (sAUC): Saliency values tend to be higher on average in the center of an image, an issue known as the "center bias". The shuffled AUC metric is a variant of the AUC that compensates for this bias by scoring a center prior at chance.

– Normalized Scanpath Saliency (NSS): The NSS metric was introduced by [Peters et al., 2005]. It measures the values of the normalized saliency map at fixation locations, then averages them together. SM is the saliency map and g corresponds to the location of one fixation.

$$NSS(g) = \frac{SM(g) - \mu_{\mathrm{SM}}}{\sigma_{\mathrm{SM}}} \tag{3.2}$$

– Similarity (SIM): The similarity metric takes as input the normalized saliency map and ground truth and computes the sum of minimum values at each pixel. The closer this sum is to 1 the higher the overlap between the two maps. In the following equation F denotes the normalized fixation map, while SM denotes the normalized predicted saliency map.

$$SIM(F, SM) = \sum min(SM_{\mathrm{i}}, F_{\mathrm{i}}) \tag{3.3}$$

– Pearsons Correlation Coefficient (CC): Also called linear correlation coefficient, is typically used in many disciplines whenever measuring the correlation between two variables. In this case, the saliency and ground truth maps are treated as random variables, so that their linear relationship may be measured.

$$CC(F, SM) = \frac{\sigma(SM, F)}{\sigma(SM)\sigma(F)} \tag{3.4}$$

3.3 Technical Aspects

The entirety of this project has been implemented in the Python programming language, with the exception of some bash scripts. All models were developed using the PyTorch library. Computation in both parts utilized 2 Graphic Processing Units (GPUs), which was particularly important for training within a reasonable amount of time (training one model under this framework required approximately 2-3 days, depending on the model and data size, as well as the resolution of the images). The first part of the project was developed on a single desktop containing 2 GPUs of

12xGB within a linux operating system. The second part was done using a SLURM device, where resources were requested on runtime (typically 2 GPUs of 11xGB for training and 1 for inference).

Chapter 4

Results

4.1 Results on Modular Network

On the first task, that is recognition of animals the sample size of each class was 5950. The dataset included 5950 animals from the families of canine and feline and 5950 "random" pictures from geological formations, flora, rock, fungi, construction sites. For validation, given that we have a decent data size we used the hold out method, which is the common practice in deep learning algorithms. 70% of the dataset was used for training and the remaining 30% was the test set. The learning progress has been plotted and is presented in figure 4.1.

Training Results on a second level task with target class Canines the pretrained layers were loaded from the previous task on all layers. 3829 samples were loaded from each class, split 80/20 between train and test set. a) Different initial learning rates were used per layer, the closer to the start the smaller the learning rate (learning rate for layer3 was 0.1, for layer2 0.065, for layer1 0.04). The initial training produced results with significant overfit, as the gap between test set and train set was huge at the point were training has plateaued (Fig. 4.2). We repeated training, this time applying image transformations on the dataset; a common practice to artificially increase the data size. More specifically we applied Random Resized Crop, which uses different segments from each image at random and Random Horizontal Flip

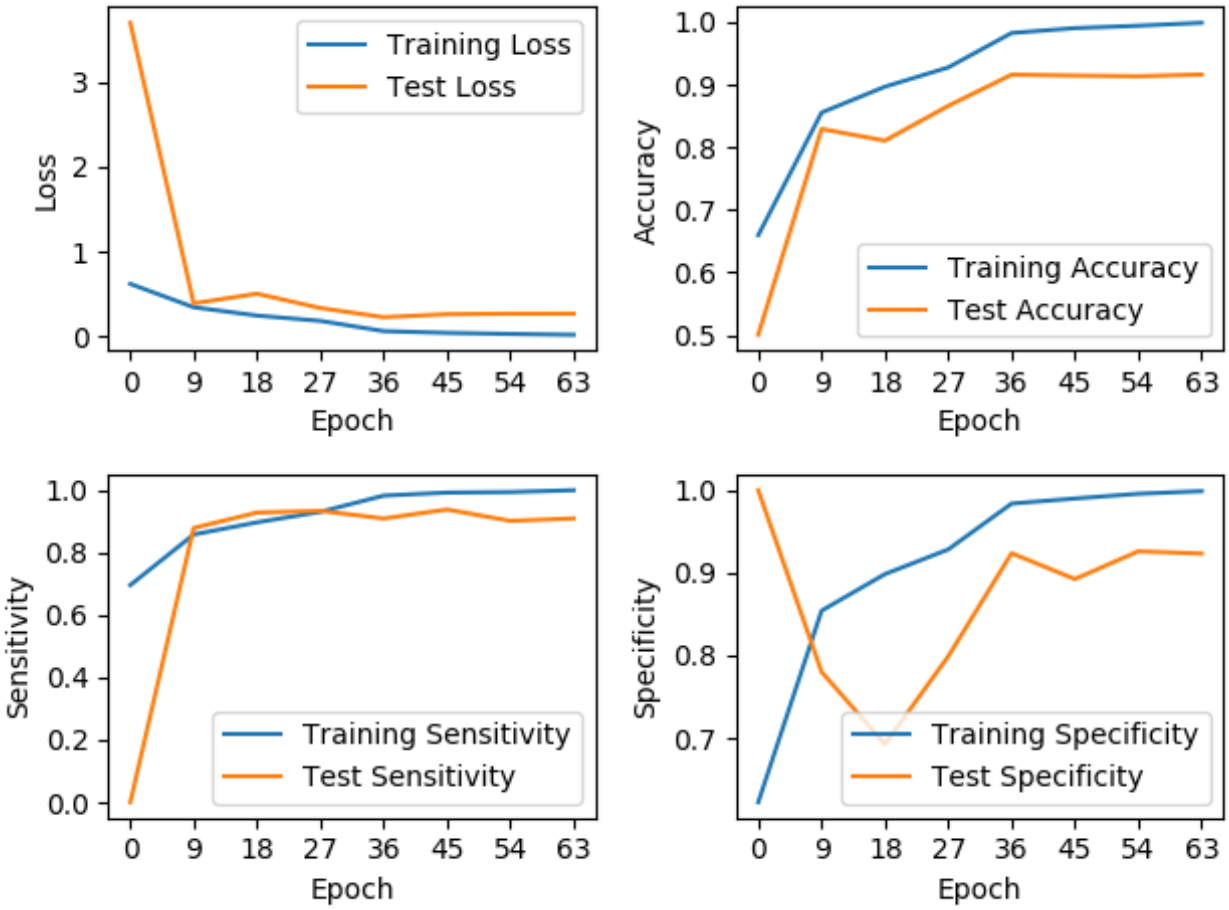


Figure 4.1: Training on Animal recognition. There is some overfit as the trained model achieves a final 0.974 accuracy on train set but 0.9 accuracy on test set.

Table 4.1: Final performance on the Animal recognition task.

	Train Set	Test Set
Accuracy	0.974	0.91
Sensitivity	0.97	0.9
Specificity	0.98	0.92

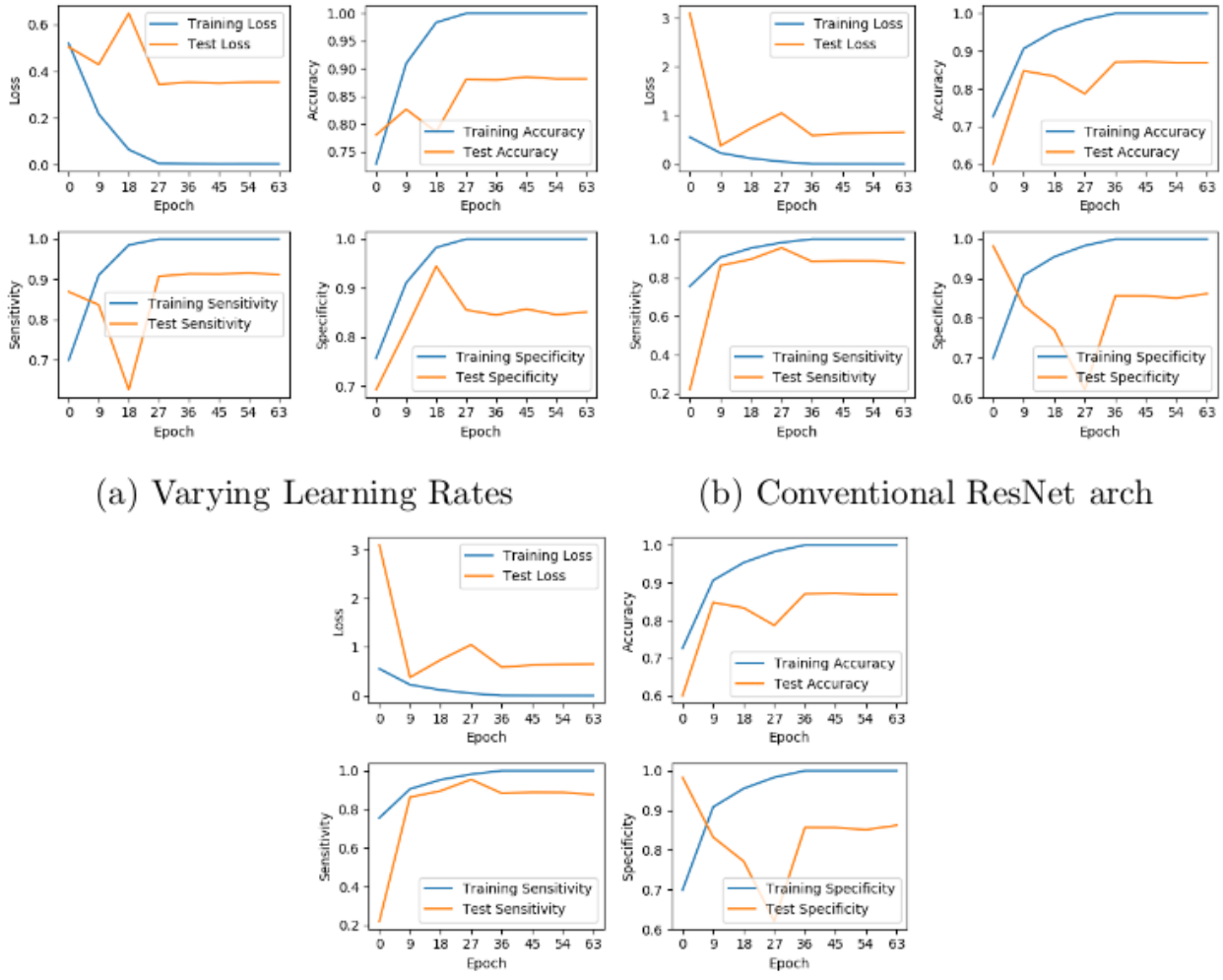
Table 4.2: Final performance on the Canine recognition task.

	Frozen Layers		Conventional		Varying Learning Rates	
	train	test	train	test	train	test
Accuracy	0.84	0.79	**0.91**	**0.87**	0.908	**0.87**
Sensitivity	0.85	0.85	**0.92**	0.866	0.908	**0.88**
Specificity	0.82	0.74	**0.9**	**0.873**	0.908	0.85

which simply flips some of the images. These transformations were applied on both train and test set, so as to artificially increase the dataset size and reduce overfit (Fig. 4.3). Training on the Frozen model was highly unstable; increasing the total epoch number was necessary, in this case (we brought it up to 135 from 30).

Results on performance for the second task are demonstrated in table 4.2. It appears that the conventional method not only has the more stable training (fig 4.3) but it outperforms our other strategies. Freezing the layers has a devastating effect in performance, as it's beaten by a margin of 7% from the conventional method both on train and test set. We believed that the varying learning rates method would improve performance by allowing the deeper layers to make bigger steps towards training rather than the pre-trained ones. However, performance was not improved; it is only very slightly worse than the conventional method on the train set and it does equally well on the test set.

After training the network on Task 2 we decided to investigate whether the original layers remember the first task, as a sort of catastrophic forgetting test. Towards this goal, the first layers of the network were loaded to predict classes from Task 1. All of the data from the previous task was gathered and the two networks whose layers were not frozen were used to predict whether they are seeing an animal or not. The fully connected layer from the task 1 module was loaded along with the convolutional

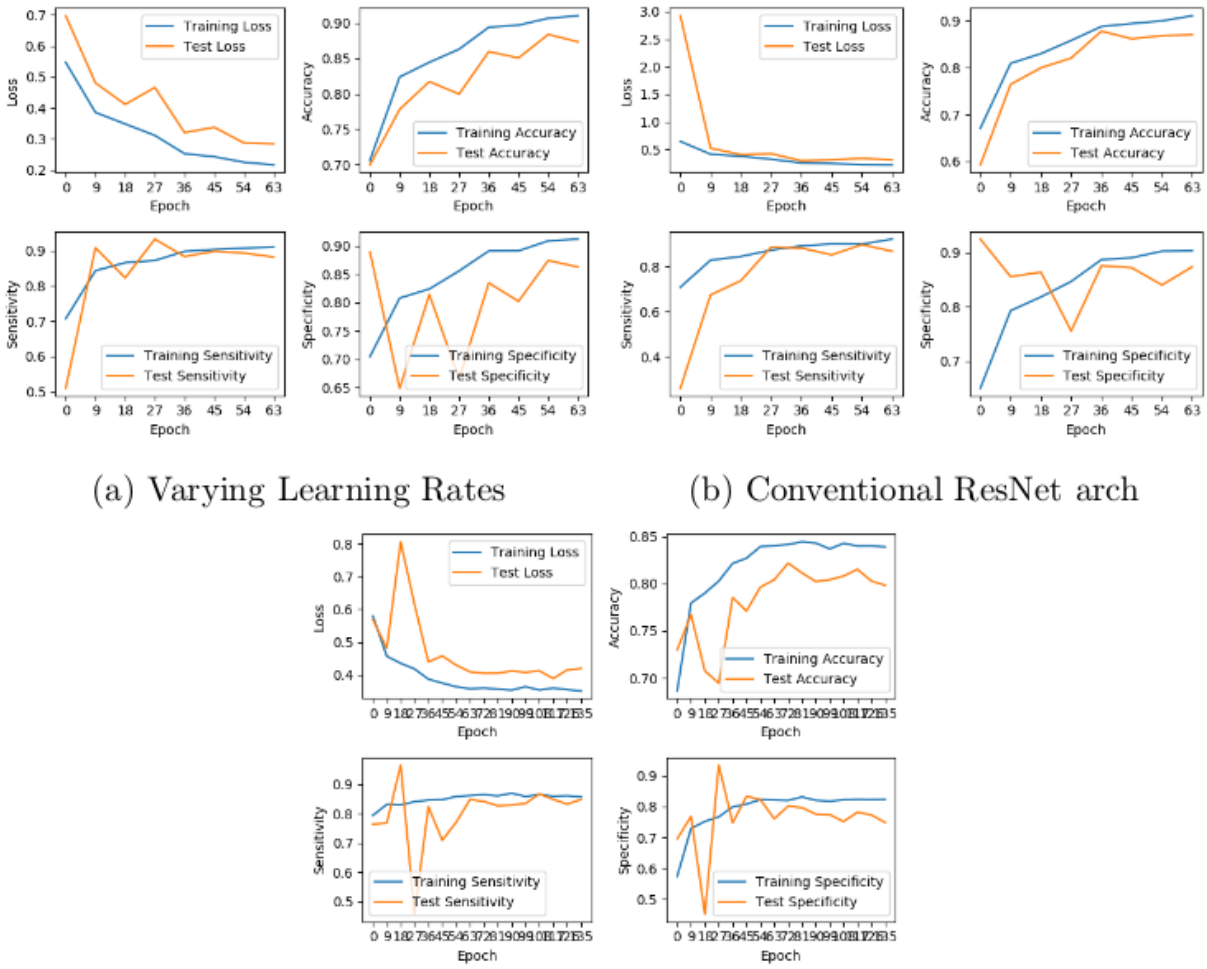


(a) Varying Learning Rates

(b) Conventional ResNet arch

(c) Frozen Shallow Layers

Figure 4.2: Initial training on Canine recognition. There is significant overfit in our initial training which prompted us to apply data manipulation techniques.



(a) Varying Learning Rates

(b) Conventional ResNet arch

(c) Frozen Shallow Layers

Figure 4.3: Final training on Canine recognition, were data was extended with preprocessing techniques. a) Different initial learning rates were used per layer, the closer to the start the smaller the learning rate (learning rate for layer 3 was 0.1, for layer 2 0.065, for layer 1 0.04). b) The conventional ResNet architecture illustrates the results of the baseline approach, that is training the whole network. c) Frozen Shallow Layers strategy pertains to freezing the part of the network that has been trained on the simple task, this part was particularly unstable so we increased the total number of epochs to 135. There is still some overfit; however, to the best of our efforts we could not further reduce this effect.

Table 4.3: Catastrophic forgetting test.

	Baseline	Conventional	VLR
Accuracy	0.974	0.81	0.75
Sensitivity	0.97	0.65	0.57
Specificity	0.98	0.96	0.94

layers that received training on the second task. Thus, we reconstructed the original model architecture while retaining the weights fine-tuned for the second task. Interestingly, decreasing the learning rates of the first layers seems to worsen forgetting of the first task compared to the conventional method. In both cases, the network that has been trained on the more complex class suffers a drop in performance compared to the baseline.

4.2 Results on Temporally-aware Saliency Network

Regarding our extensions to the state of the art saliency predicting model, we assessed the performance firstly on the same DHF1K dataset [Wang et al., 2018], which is the dataset upon which the conv and ConvLSTM layers were trained. This way, we could assess the quality of our baseline with respect to the state of the art in video saliency prediction. The descending loss of the ConvLSTM has been plotted to demonstrate learning progress 4.4

Afterwards, the model was assessed on the newly released EgoMon dataset to draw our conclusions in the egocentric domain. We adopt standard saliency metrics when assessing our results on DHF1K dataset, and focus in NSS in the case of EgoMon: Normalized Scanpath Saliency (NSS), Similarity Metric (SIM), Linear Correlation Coefficient (CC), AUC-Judd (AUC-J), and shuffled AUC (s-AUC) [Bylinskii et al., 2016]. In the case of the egocentric datasets though, evaluation is more challenging as there is one gaze pixel per frame instead of a saliency heat map. For this and some other technical reason we restricted our evaluation to just the NSS metric

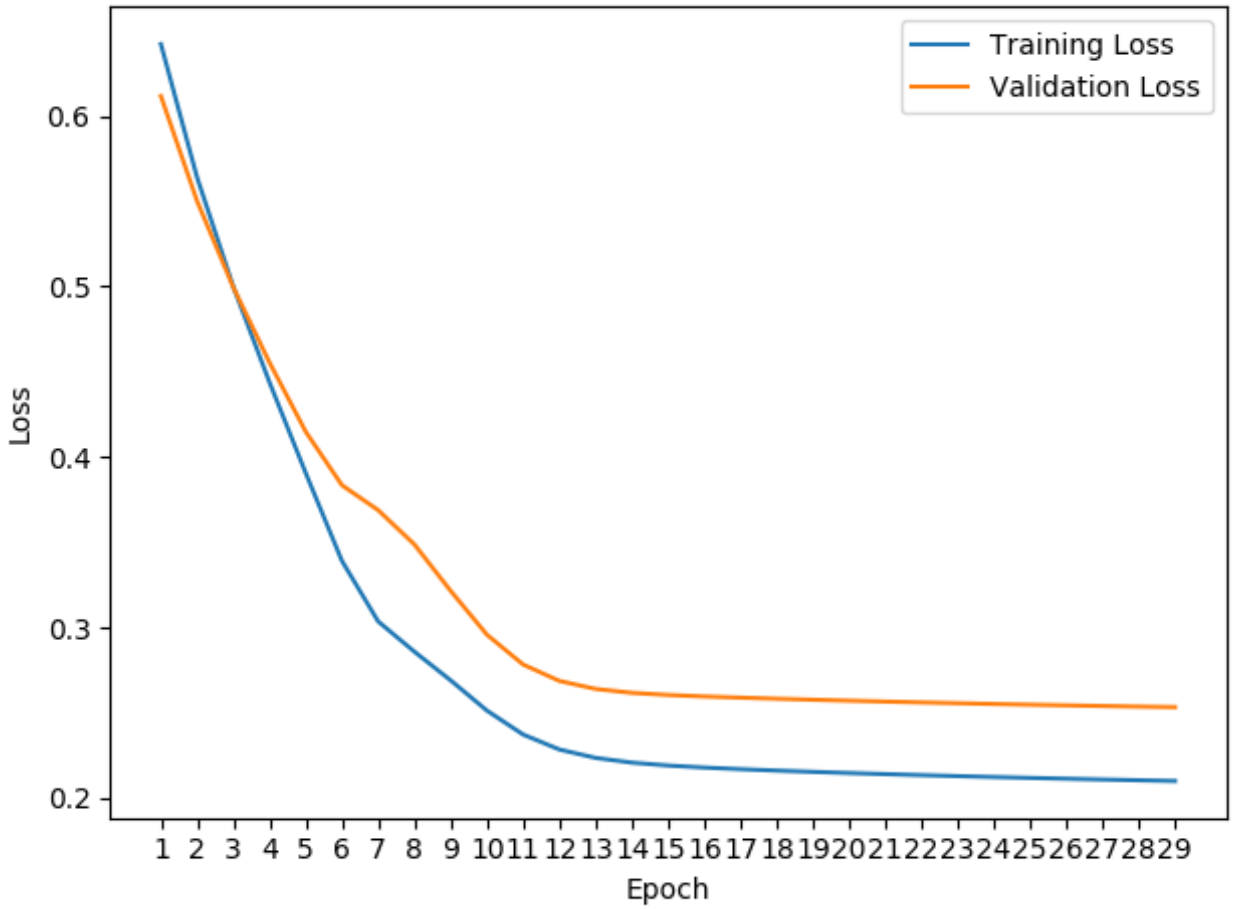


Figure 4.4: Training of the ConvLSTM module. The descending loss indicates that the network is learning.

Table 4.4: Performance on the DHF1K dataset.

	AUC-J ↑	sAUC ↑	NSS ↑	CC ↑	SIM ↑
SoA [Wang et al., 2018]	0.885	0.553	2.259	**0.415**	**0.311**
SalGAN [Pan et al., 2017]	**0.930**	**0.834**	**2.468**	0.372	0.264
+ conv	0.743	0.723	2.208	0.303	0.261
+ConvLSTM	0.744	0.722	2.246	0.302	0.260

Table 4.5: NSS metric across the DHF1K and EgoMon datasets.

	DHF1K	EgoMon
SalGAN [Pan et al., 2017]	**2.468**	**2.079**
+conv	2.208	1.250
+ConvLSTM	2.246	1.247

[Peters et al., 2005].

Table 4.4 indicates that, surprisingly, the off-the-shelf (frame-based) SalGAN model [Pan et al., 2017] outperformed the state of the art model on the DHF1K [Wang et al., 2018] dataset. On the other hand, the quality of the prediction decreases when the conv or ConvLSTM layers are trained on top, which indicates that the domain adaptation is damaging the performance of the original SalGAN.

Table 4.5 indicates an even worse loss of performance when adding this adaptation layers in the EgoMon dataset. Nevertheless, the more detailed look provided in Table 4.6 shows that the off-the-shelf model performs better during free-viewing recordings, where saliency arises according to the intrinsic visual characteristics of a scene (bottom-up). Notice that DHF1K fixations were mostly recorded in a free-

Table 4.6: Performance on different EgoMon tasks (NSS metric).

	free-viewing recordings (bottom-up saliency)				
	bus ride	botanical gardens	dcu park	walking office	AVERAGE
SalGAN [Pan et al., 2017]	**1.618**	**1.182**	**4.374**	**3.435**	**2.652**
+ conv	0.947	0.846	0.683	0.745	0.805
+ ConvLSTM	0.827	0.576	1.172	1.040	0.904
	task-driven recordings (top-down saliency)				
	playing cards	presentation	tortilla		AVERAGE
SalGAN [Pan et al., 2017]	0.967	1.360	1.618		1.315
+ conv	1.114	**1.966**	2.002		1.694
+ ConvLSTM	**1.141**	1.897	**2.077**		**1.705**

viewing manner as well. On the other hand, the adaptation layers are beneficial in those scenes where the user is engaged in an activity, in which task-driven saliency (top-down) [Murabito et al., 2017] dominates.

We observe that the temporal saliency adaptation improves performance when the viewer is engaged in a task and with a narrow field of view, but, on the other hand, losses are measured when the viewer is simply free-viewing an open scene. Encouraged by these results, we have computed the saliency maps pertaining to the Epic Kitchens object detection challenge [Damen et al., 2018]. We believe that these data can be valuable for third-party research focusing on other task such as object detection [Reyes et al., 2016] or video summarization [Xu et al., 2015].

We illustrate some frames randomly picked from videos of our datasets on figures 4.5 and 4.6 for qualitative analysis. The saliency maps show that our conv and ConvLSTM extensions reinforce the higher probability pixels at the expense of darkening the lower ones. This effect seems to be beneficial in the case of task-driven activities, as the scene tends to be constant in time and the region of interest is fixed in space. However, free-viewing tasks contain changing scenes with sparser saliency maps.

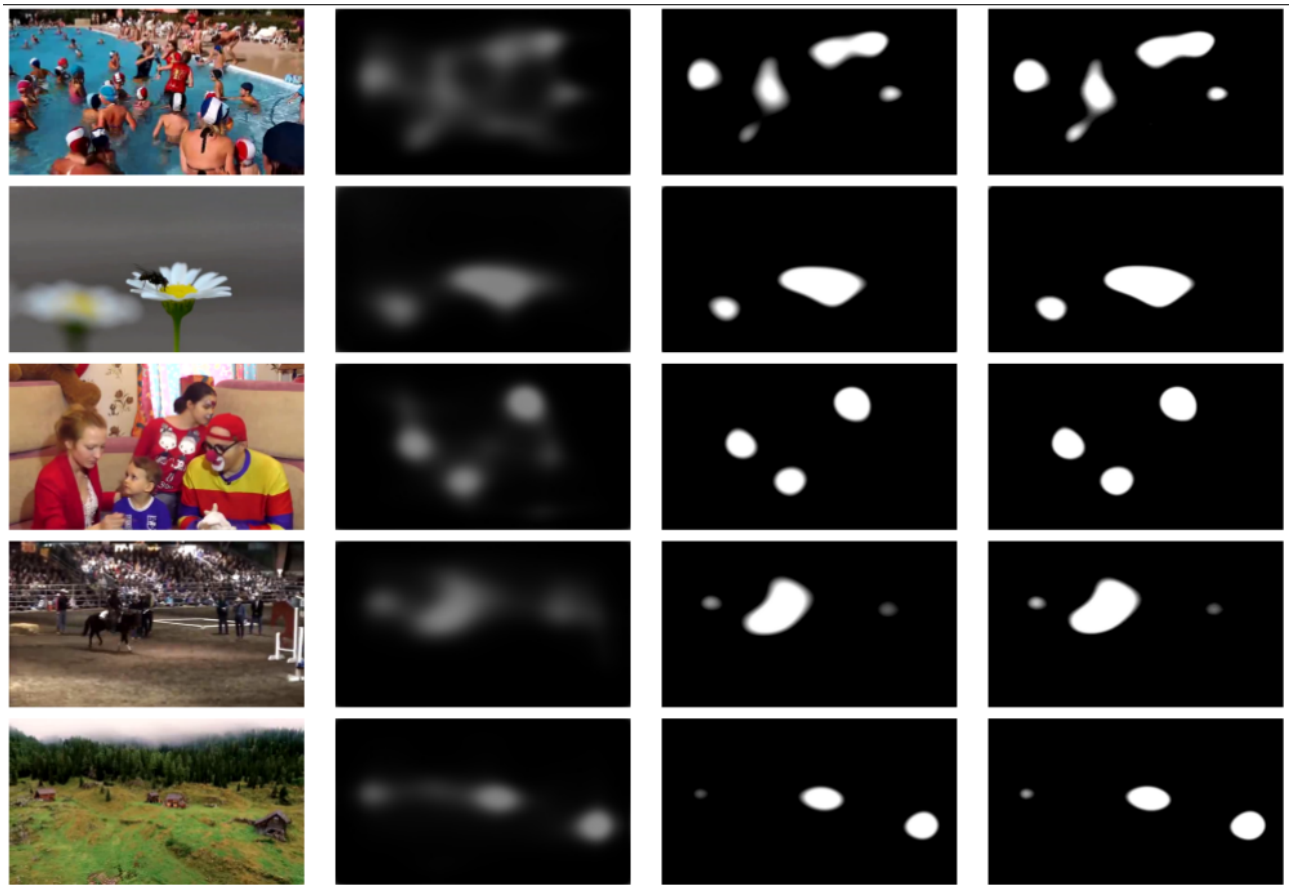
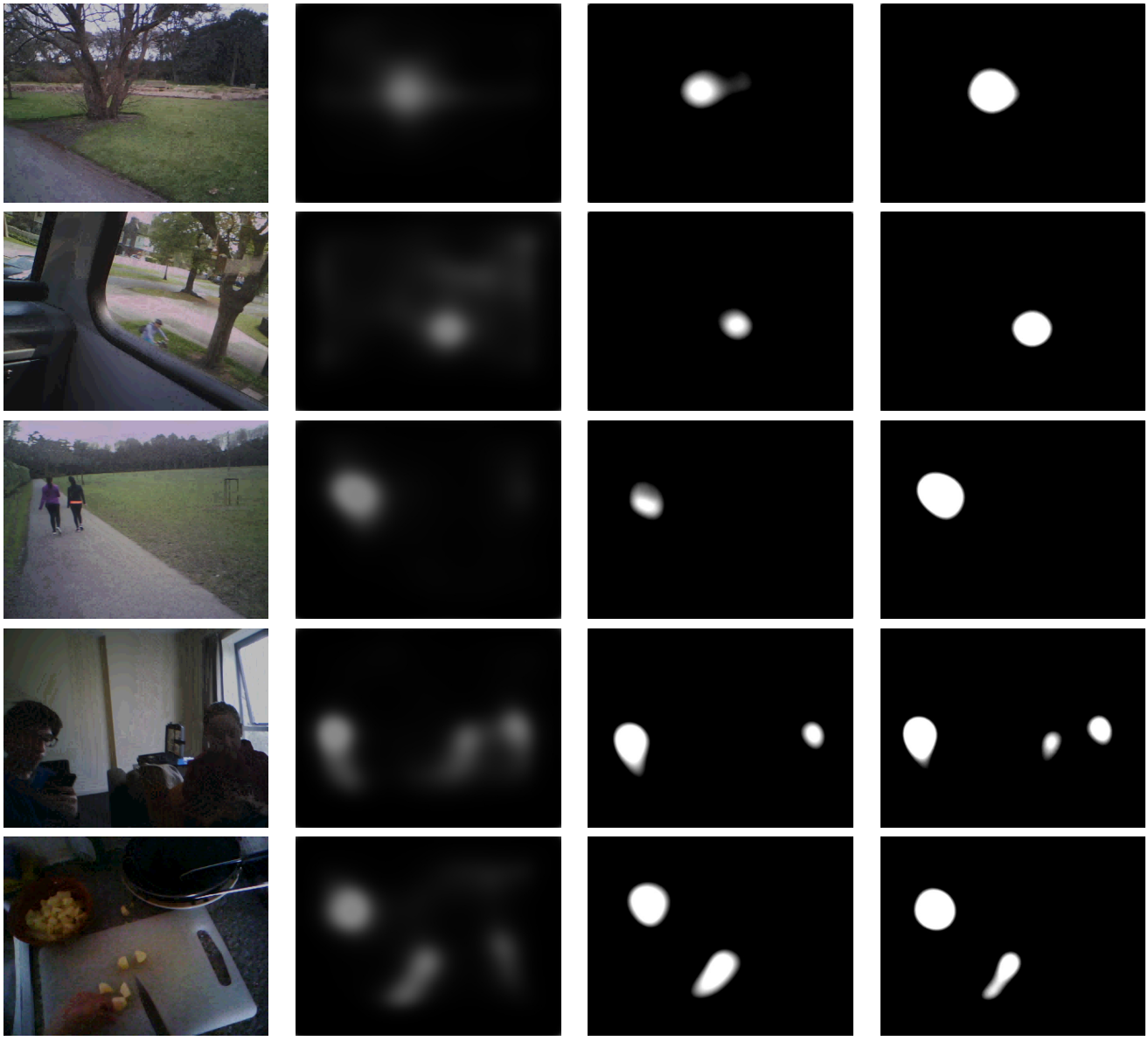


Figure 4.5: DHF1K Qualitative Analysis. From left to right: a random frame from a video, a saliency map predicted by SalGAN, a saliency map predicted by ConvLSTM, a saliency map predicted by simple convolutional.

Figure 4.6: EgoMon Qualitative Analysis. From left to right: a random frame from a video, a saliency map predicted by SalGAN, a saliency map predicted by ConvLSTM, a saliency map predicted by simple convolutional. The 3 first rows refer to free-viewing activities, while the two last one are task-driven.



Chapter 5

Conclusions and future work

In this work, we demonstrated how using the brain as a guide may support further development of deep learning architectures. We saw how catastrophic forgetting is highly prevalent in transfer learning settings, even when the preceding class is but a superclass of the lastly learned one. Furthermore, we deduced that varying the learning rates yields no better results and may even worsen performance, illustrating that there is no clear analogue with brain spiking activity and the rate by which networks learn. We also worked to improve current methods on predicting saliency using datasets from humans observing and even interacting with their environment. Our model uses a simple grayscale heatmap output from another state of the art model to learn temporal patterns. In this challenging setting the model had a hard time beating the aforementioned state of the art model. Nevertheless it shows a promising boost on task-driven settings, where we speculate that temporal patterns are more prevalent than the seemingly random free-viewing of scenes.

We estimate that it will be highly fruitful to allow the saliency model access to more information than just the heat map produced by another model. An interesting and rather straightforward approach would be to use a deeper representation of the state of the art model but not its final 1-dimensional output, as the deeper representation contains a higher dimensionality of information. Another potential idea would be to simply intensify the salient features in the original raw images using the saliency

map from SalGAN as a guide. Furthermore modular networks such as the one we implemented would have a lot to gain by such soft-attention guides. Soft-attention could be used for the different modules of the network to focus on increasingly minute details that separate complex classes. On top of that, these attention guides may assist in allowing the network to remember the most significant information from any task it has previously learned; therefore, tackling the problem of catastrophic forgetting. Finally, an analogue of ImageNet in the video domain, much less in the egocentric domain, does not currently exist with the same hierarchical structure. We speculate that datasets of this nature could further propagate research in this direction.

Bibliography

[Bak et al., 2017] Bak, C., Kocak, A., Erdem, E., and Erdem, A. (2017). Spatio-temporal saliency networks for dynamic saliency prediction. *IEEE Transactions on Multimedia*.

[Borji, 2012] Borji, A. (2012). Boosting bottom-up and top-down visual features for saliency estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Buschman and Miller, 2007] Buschman, T. J. and Miller, E. K. (2007). Top-down versus bottom-up control of attention in the prefrontal and posterior parietal cortices. *science*, 315(5820):1860–1862.

[Bylinskii et al., 2016] Bylinskii, Z., Judd, T., Oliva, A., Torralba, A., and Durand, F. (2016). What do different evaluation metrics tell us about saliency models?

[Bylinskii et al., 2018] Bylinskii, Z., Judd, T., Oliva, A., Torralba, A., and Durand, F. (2018). What do different evaluation metrics tell us about saliency models? *IEEE transactions on pattern analysis and machine intelligence*.

[Damen et al., 2018] Damen, D., Doughty, H., Farinella, G. M., Fidler, S., Furnari, A., Kazakos, E., Moltisanti, D., Munro, J., Perrett, T., Price, W., and Wray, M. (2018). Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *conference on Computer Vision and Pattern Recognition*.

[Essen et al., 1992] Essen, D. V., Anderson, C., and Felleman, D. (1992). Information processing in the primate visual system: an integrated systems perspective. *Science Magazine*.

[Farooq, 2000] Farooq, A. (2000). Biologically inspired modular neural networks. *PhD Dissertation, Virginia Tech*.

[Fathi et al., 2012] Fathi, A., Li, Y., and Rehg, J. M. (2012). Learning to recognize daily actions using gaze. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7572 LNCS(PART 1):314–327.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

[Goodfellow et al., 2013] Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

[Gorji and Clark, 2018] Gorji, S. and Clark, J. J. (2018). Going from image to video saliency: Augmenting image salience with dynamic attentional push. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7501–7511.

[Graves et al., 2013] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

[H. and N., 1962] H., H. D. and N., W. T. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J Physiol*.

[Harel et al., 2006] Harel, J., Koch, C., and Perona, P. (2006). Graph-based visual saliency. In *Neural Information Processing Systems (NIPS)*.

[Hassabis et al., 2017] Hassabis, D., Kumaran, D., Summerfield, C., and Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron Review*.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Huang et al., 2015] Huang, X., Shen, C., Boix, X., and Zhao, Q. (2015). Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks. In *IEEE International Conference on Computer Vision (ICCV)*.

[Huang et al., 2018] Huang, Y., Cai, M., Li, Z., and Sato, Y. (2018). Predicting gaze in egocentric video by learning task-dependent attention transition. *arXiv preprint arXiv:1803.09125*.

[Hubel and Wiesel, 1962] Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.

[Itti and Koch, 2001] Itti, L. and Koch, C. (2001). Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3):194.

[Itti et al., 1998] Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (20):1254–1259.

[Jiang et al., 2015] Jiang, M., Huang, S., Duan, J., and Zhao, Q. (2015). Salicon: Saliency in context. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1072–1080.

[Judd et al., 2009] Judd, T., Ehinger, K., Durand, F., and Torralba, A. (2009). Learning to predict where humans look. In *IEEE International Conference on Computer Vision (ICCV)*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[LeCun et al., 1998a] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[LeCun et al., 1998b] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998b). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Linardos et al., 2018] Linardos, P., Mohedano, E., Cherto, M., Gurrin, C., and Giro-i Nieto, X. (2018). Temporal saliency adaptation in egocentric videos. *arXiv preprint arXiv:1808.09559*.

[Marblestone et al., 2016] Marblestone, A. H., Wayne, G., and Kording, K. P. (2016). Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*.

[Murabito et al., 2017] Murabito, F., Spampinato, C., Palazzo, S., Pogorelov, K., and Riegler, M. (2017). Top-Down Saliency Detection Driven by Visual Classification.

[Pan et al., 2017] Pan, J., Ferrer, C. C., McGuinness, K., O'Connor, N. E., Torres, J., Sayrol, E., and Giro-i Nieto, X. (2017). SalGAN: Visual Saliency Prediction with Generative Adversarial Networks.

[Peters et al., 2005] Peters, R. J., Iyer, A., Itti, L., and Koch, C. (2005). Components of bottom-up gaze allocation in natural images. *Vision research*, 45(18):2397–2416.

[Ptak, 2012] Ptak, R. (2012). The frontoparietal attention network of the human brain: action, saliency, and a priority map of the environment. *The Neuroscientist*,

18(5):502–515.

[Reyes et al., 2016] Reyes, C., Mohedano, E., McGuinness, K., O'Connor, N. E., and Giro-i Nieto, X. (2016). Where is my phone?: Personal object retrieval from egocentric images. In *Proceedings of the first Workshop on Lifelogging Tools and Applications*, pages 55–62. ACM.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

[Shi et al., 2015] Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., and Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, abs/1506.04214.

[Simonyan and Zisserman, 2014a] Simonyan, K. and Zisserman, A. (2014a). Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576.

[Simonyan and Zisserman, 2014b] Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Su and Grauman, 2016] Su, Y.-C. and Grauman, K. (2016). Detecting engagement in egocentric video. In *European Conference on Computer Vision*, pages 454–471. Springer.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[Theis et al., 2018] Theis, L., Korshunova, I., Tejani, A., and Huszár, F. (2018). Faster gaze prediction with dense networks and fisher pruning. *arXiv preprint arXiv:1801.05787*.

[Wang et al., 2018] Wang, W., Shen, J., Guo, F., Cheng, M.-M., and Borji, A. (2018). Revisiting video saliency: A large-scale benchmark and a new model.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4894–4903.

[Xu et al., 2015] Xu, J., Mukherjee, L., Li, Y., Warner, J., Rehg, J. M., and Singh, V. (2015). Gaze-enabled egocentric video summarization via constrained submodular maximization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2235–2244.

[Zhu and Xu, 2018] Zhu, S. and Xu, Z. (2018). Spatiotemporal visual saliency guided perceptual high efficiency video coding with neural network. *Neurocomputing*, 275:511–522.