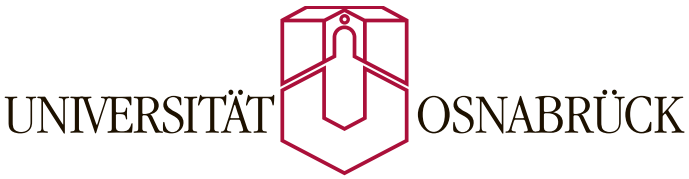

SaccadicEyeMovementNET

Submitted by : Marlon Dammann , Nils Niehaus, Argha Sarker

Matriculation : 982861,983301,990398

Course : Implementing Artificial Neural Networks with Tensorflow

Tutor : Leon Schmidt

Date : 7 April 2022

1 Introduction

Human vision is a fascinating ability. It makes us capable of interpreting our surroundings to interact safely and accurately with little conscious effort.[1] How humans perceive information from an image has been the point of focus in scientific studies for a long time. In this project, we investigate and try to replicate the mechanism of human interaction with an image, specifically how the image and its contents are being classified.

However, instead of simply feeding images into a basic classification model, we take a modern deep learning approach with the intention to classify an image based on potential focus points of human eyes represented by small image patches from a large image via reinforcement learning methods. So the training objectives amount to classifying the current patch as well as obtaining the policy for deciding on the next focus points. The model is supposed to represent the saccadic movement of the human eye.

2 Motivation

With advanced, artificial intelligence techniques, such as deep learning and reinforcement learning, machines can predict and do things similar to humans. Perceiving visual information about our surroundings with so little information taken as input and cleverly chosen focus points of how to navigate the scenario to extract most information with less available data is a fascinating question that deep learning techniques can recently answer. Visual Perception depends critically on frequent changes in the scene. Usually, our view of the world is continuously changed by Saccades that continue to move the eyes abruptly over a fraction of a degree of visual observations.[2] As a consequence of these several sorts of eye movements, our point of view changes more or less continually.[3]

How to cope with the rapidly changing environment and how to train a system that can precise a visual environment and learn to focus on a different portion of the visual environment and learns new information from this continuously changing environment. To answer these questions we take two different paths that answer individual problems. There are other image classification techniques such as SVM, K-neighbour, texture-based approach, etc.[7] While deep convolutional neural networks (CNNs) have shown great success in single-label image classification as it is efficient and shows significant improvements compare to other methods, we use this approach to learn the information from different focus points for extracting the features from image data and classifying the images based on the various focus points.[6]

Furthermore, navigating the visual environment and selecting the focus

points can be tricky and the key to perceiving the visual environment successfully. The network has to learn information from the image patches and store it over extended time intervals, we chose RNN(Recurrent Neural Network) techniques like LSTM(Long Short-Term Memory), as it can learn over a time step with recurrent backpropagation. Which makes it an ideal choice of architecture for learning the sequence can perceive the visual information from the whole image [7]. From here we chose reinforcement learning methods to take action based on the output of the network and based on the reward the agent receives it selects its next focus point accordingly.

3 Methodology

3.1 Analysis of the task

As stated in the motivation section, the elected task is to classify images not in the standard paradigm of Deep CNNs, e.g., using the entire image as model input, but instead as a sequence of image patches; like several focus points over time from a human eye looking at the image.

To achieve this within a Deep Learning framework, we need to define a model which is capable of both classifications and deciding on the next image patch to select for input. The initial idea was to select an image patch of size NxN from the center of the image as model input and to then output both a softmax-based classification and the coordinates from where to sample the next image patch.

Ideally the classification is done via target labels from a supervised dataset whilst the movement of the focus point is done by applying a reinforcement algorithm to reward a good policy for selecting the next point of interest in an image. In the next subsections the choices regarding the dataset, model, training, and the general setting as a Markov Decision Process (MDP) are elaborated on.

3.2 Dataset choice

By recommendation, and since it is a fitting choice, we decided to use MS Coco as our dataset for training. In MS Coco, henceforth only COCO, we have a dataset which is originally designed for object detection tasks [5]. Coco has 80 represented classes, ranging from the most common class âpersonâ to fewer common ones like âhot dogâ. The main advantage in using COCO over other labelled datasets is that COCO contains higher resolution images which means that it is easier to select image patches from the images which are still

meaningful. In another dataset like CIFAR10 we would have images which are already too small to select patches from with meaningful content.

Since images in COCO often contain multiple classes, the classification with a single label is not always reasonable, however, using the largest object in the image as the single label for the image should still yield good results in a classification task. For modifying COCO, we decided to download all the images in COCO from eleven selected classes: person, car, chair, bench, truck, bottle, cup, bowl, bicycle, book, and clock. Initially it appeared as if the method we were using, namely the methods from the COCO dataset module for python, were already splitting the images into folders with a single label, according to the class of the largest object in the image. Later, after inspecting the folders, it became clear that we downloaded about ten percent duplicate images which were present multiple times in different classes. Since our project run short on time the solution was a function to remove duplicate images, however, the downloading function was not updated to only download non-duplicates, leaving it somewhat inefficient.

For the preprocessing of the dataset, we decided to normalize the values of the image pixels to the range between zero and one and to resize all images to 400x400 pixels, regardless of aspect ratio and original size. For the batch size we chose 64, the class target was accordingly preprocessed to appear as a one-hot vector. Additionally, it was decided to use 32x32 pixels as the size of our image patches to be extracted from the image.

The motivation behind the image patch size was the inspection of the dataset images to find an appropriate size of an image patch which is big enough to include some features and small enough to not include too much. The image reshaping to 400x400 pixels was decided upon since the alternative of zero-padding the images to the size of the biggest image aspects present in the data seemed less promising overall. The hypothesis was that a uniform shape for all images should result in a much more accurate policy for looking for the next patch. This will be further elaborated upon in the coming sections.

3.3 Initial model choice

The initial idea regarding model choice was to build a model with some features of a common CNN, some features of a RNN and an output that accommodates both the classification part and the selecting of the next image patch. Initially it was planned to use one model for doing both. For this initially four building blocks were decided upon, firstly, a feature extractor as a set of Convolutional Layers with some Pooling to create a feature representation. Secondly, this

feature representation would be the input to a LSTM cell or layer which then feeds into two model âheadsâ called policy head and classifier head. The classifier head should consist of a wide dense layer followed by a dense layer of the same width as our classes with a Softmax activation function. The policy head should consist of a wide dense layer followed by a dense layer of size two with linear activation for producing the coordinates of where to look next relative to the current position.

However, this initial idea was scrapped and instead two different models were used, one is the SaccadicNet Classifier (SN Classifier) the other is the SaccadicNet Eye (SN Eye). Both models have the same structure as the initial model plan up except for the SN Eye which has a different last layer for the output. Figure 1 shows both models with selected parameters.

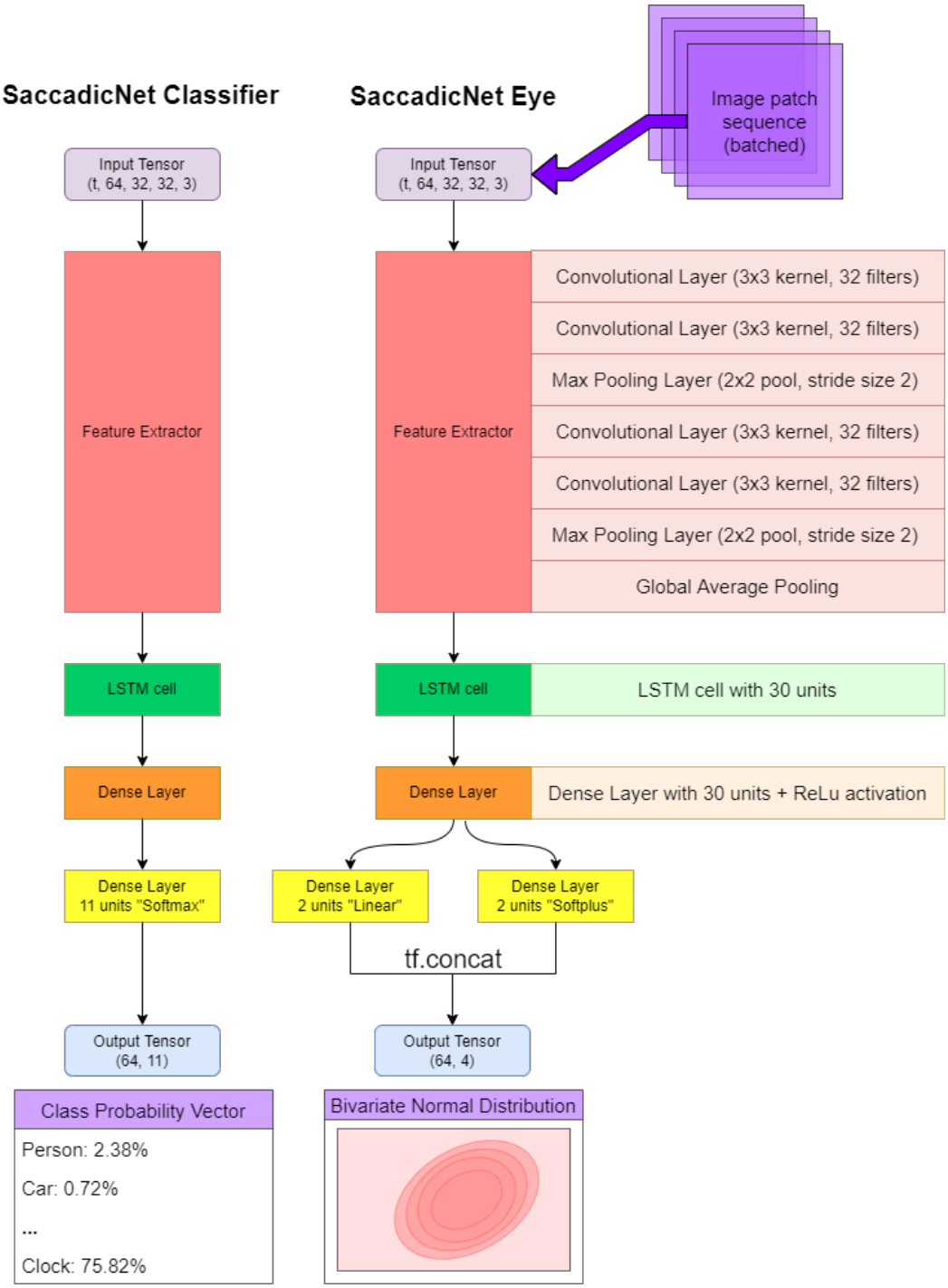

Figure 1: Model Architecture

The direction of the SN Eye model part was shifted to be non-deterministic. Contrary to the previous design the output now consists of the parameters for a Bivariate Normal Distribution in the form of two values for the mean, produced by a dense layer with linear activation, and two values for the covariance, produced by a Dense Layer with Softplus activation. Since the image patch selecting functions were later defined and only the range from 0-400 was reasonable for the mean it would also make sense to instead use a single Dense

Layer with Softplus activation for all four values.

The model would now not produce a deterministic value of where to take the next image patch from but rather a probabilistic value which should resemble the human perception more but also enable for more exploration during training, possibly leading to better results.

3.4 Training

For the training it was necessary to first write required environmental functions which represent states and actions in the environment. The environment here is a mixture of the image and the model output. The sequence needs to be fetched from the images and fed into both models. To do this an Environment class was written which can fetch an image patch given coordinates and to sample coordinates from the Bivariate Normal Distribution. Since a terminal state needs to be defined it was decided that the maximum length of the image patch sequence must not exceed 20 image patches.

One training episode on a batch of images now consists of generating a sequence of image patches and storing the output of the model for each step in the sequence. The initial sequence element, the first image patch, is always the image patch from the center of the image. Therefore, each episode the SN Eye can receive losses for 19 steps in the sequence whereas the SN Classifier can receive 20 losses averaged over the batch and the sequence steps. The loss for the SN Classifier is defined as the Categorical Cross-Entropy Loss (CCE loss), the loss for the SN Eye is customized. Within the GradientTape context of TensorFlow the CCE loss of a batch at each sequence step was taken and then averaged. With this loss the gradients were calculated and applied, using the Adam Optimizer.

The SN Eye model was trained based on REINFORCE, however, during implementation a problem was encountered which could not be fixed within the available time scope. The desired and, partially also implemented, procedure is elaborated on further.

The idea of the REINFORCE algorithm is to use sampling for each episode to derive Policy Gradients[8]. The sampling is done by generating the sequence of image patches. The 19 last image patches here were patches collected at a sampled coordinate which is based on the Bivariate Normal Distribution which is the output of the SN Eye model. How good or bad a generated distribution is can only be evaluated by the performance of the classifier improving given a specific policy.

A policy within the framework of our problem is a distribution of probabilities over all possible actions we can take. An action is a sampled coordinate, meaning that the sum of all possible coordinates is our action space. The Bivariate Normal Distribution represents the probability of each action. Importantly, since it is possible to sample outside the scope of our 400x400 image, coordinates for which the image patch would include pixels outside the image are set to a minimum or maximum value within the scope of the image. Therefore, the probability of actions which are illegal are added to their corresponding limit-adjusted coordinate pair at the edge of the image.

Since both models receive the exact same input sequence the Loss of the SN Classifier can be used to define a reward for the SN Eye model. The inverted CCE loss of the classifier, meaning the CCE loss to the power of minus one, with the fringe case of the loss being zero taken care of, is our reward for an action in the state, our value function. With this the sample trajectory policy gradient can be calculated for each step. The parameter update for each step is defined as, following the description of Lillian Weng [8]

$$\alpha\gamma^t G_{t\Theta} ln\pi_\Theta(A_t|S_t)$$

In the case of the SN Eye model the formula can be modified by these steps.

1. Items are numbered automatically.
2. The reward for the discounted future reward can be substituted by the inverted CCE loss at the specified step in the sequence with the condition that the Loss must be greater than zero.
3. The log probability of the action given the state can be substituted by probability density function of the distribution at step t for the sampled coordinate at step t.
4. An entropy-based measure is added to ensure that the covariance is kept at a certain range and the model is punished for trying to eradicate the uncertainty too much.

Ideally after these changes the loss for the SN Eye model can be calculated and backpropagated, however, this is where the implementation failed since it was unable to use auto differentiation in GradientTape to calculate the gradient. Several attempts were made to fix this error, but all were unsuccessful.

Ideally the gradient for the update of the SN Eye model would then be the mean of the gradients for all 19 steps in the sequence. Again, the first step is excluded since the initial image patch is always defined as the center patch.

4 Results

Since the training was unsuccessful there is no report on the model performance or learning success available.

5 Evaluation

Since the training was unsuccessful there is no report on the model performance or learning success available.

6 Conclusion

In the Project SaccadicEyeMovementNET, we designed two different newwork for our task. SaccadicNetClassifier, which is responsible for the image classification and SaccadicNetEye for immitating the saccadic movement of the eye. With our limited knowledge and experience, we failed to implement the Policy Gradient Algorithm. Since we could not calculate the Gradient and calculate the loss, we do not have results from out SaccadicEyeMovementNET and we failed to evaluate the network and could not reach any conclusion. Considering the difficulty level of the project and time constrains, we are concluding the project here with out being able to train the model and derive model performance for now.

7 References

1. What is visual perception? The Interaction Design Foundation. (n.d.). Retrieved April 6, 2022, From:

   https://www.interaction-design.org /literature/topics/visual-perception/

2. Purves D, Augustine GJ, Fitzpatrick D, et al., editors. Neuroscience. 2nd edition. Sunderland (MA): Sinauer Associates; 2001. Types of Eye Movements and Their Functions. Available from: https://www.ncbi.nlm.nih.gov /books/NBK10991/

3. Coppola D., Purves D. The extraordinarily rapid disappearance of entoptic images. Proc. Natl. Acad. Sci. USA. (1996);96:8001â8003.

4. M. Jogin, Mohana, M. S. Madhulika, G. D. Divya, R. K. Meghana and S. Apoorva, "Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning," 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT), 2018, pp. 2319-2323, DOI: 10.1109/RTEICT42901.2018.9012507.

5. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

6. Amerini, I., Li, C. T., Caldelli, R. (2019). Social network identification through image classification with CNN. IEEE access, 7, 35264-35273.

7. through image classification with CNN. IEEE access, 7, 35264-35273. S. S. Nath, G. Mishra, J. Kar, S. Chakraborty and N. Dey, "A survey of image classification methods and techniques," 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014, pp. 554-557, doi: 10.1109/ICCICCT.2014.6993023.

8. Weng, L. (2018, 8 april). Policy Gradient Algorithms. LilâLog. Geraadpleegd op 8 april 2022, van https://lilianweng.github.io/posts/2018-04-08-policy-gradient/