VIDEO SALIENCY PREDICTION WITH DEEP NEURAL NETWORKS

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Juan José Nieto Salas

In partial fulfilment

of the requirements for the degree in

TELEMATIC SYSTEMS ENGINEERING

Advisors: Eva Mohedano, Kevin McGuiness and Xavier Giró

Barcelona, January 2019

Abstract

Saliency prediction is a topic undergoing intense study in computer vision with a broad range of applications. It consists in predicting where the attention is going to be received in an image or a video by a human.

Our work is based on a deep neural network named SalGAN, which was trained on a saliency annotated dataset of static images. In this thesis we investigate different approaches for extending SalGAN to the video domain. To this end, we investigate the recently proposed saliency annotated video dataset DHF1K to train and evaluate our models. The obtained results indicate that techniques such as depth estimation or coordconv can effectively be used as additional modalities to enhance the saliency prediction of static images obtained with SalGAN, achieving encouraging results in the DHF1K benchmark. Our work is based on pytorch and it is publicly available here: https://github.com/juanjo3ns/SalBCE

Resum

La predicció de saliencia és un subàmbit de la visió per computador que actualment està sent molt estudiat i que té un ampli espectre d'aplicacions. Consisteix en predir on els humans fixaran l'atenció en fotografies i videos.

El nostre treball està basat en una xarxa neuronal profunda anomenada SalGAN, la qual va ser entrenada en un conjunt de dades d'imatges estàtiques. En aquesta tesi s'investiguen diferents estratègies per extendre SalGAN al domini del video. Per poder entrenar i avaluar els nostres models s'utilitzen conjunts de videos que han sigut annotats amb saliencia, com el DHF1K. Els resultats obtinguts indiquen que tècniques com l'estimació de la profunditat o coordconv poden ser afegides per millorar la predicció de saliencia que s'obtenia amb SalGAN, aconseguint així, resultats alentadors en estàndards de referencia com el DHF1K. El nostre treball està basat en pytorch i està publicament disponible aquí: https://github.com/juanjo3ns/SalBCE

Resumen

La predicción de saliencia es un subámbito de la visión por computador que actualmente está siendo muy estudiada y que tiene un amplio espectro de aplicaciones. Consiste en predecir donde fijarán la atención los humanos en fotografías y videos.

Nuestro trabajo está basado en una red neuronal profunda llamada SalGAN, la cual fue entrenada en un conjunto de datos de imágenes estáticas. En esta tesis se investigan diferentes estrategias para extender SalGAN al dominio del video. Para poder entrenar y evaluar nuestros modelos se utilizan conjuntos de videos que han sido anotados con saliencia, como el DHF1K. Los resultados obtenidos indican que técnicas como la estimación de profundidad o coordconv pueden ser añadidas para mejorar la predicción de saliencia que se obtenía con SalGAN, consiguiendo así, resultados alentadores en estándares de referencia como el DHF1K. Nuestro trabajo está basado en pytorch y está publicamente disponible aquí: https://github.com/juanjo3ns/SalBCE

Acknowledgements

I would like to thank my supervisors Eva Mohedano, Xavi Giró and Kevin McGuiness to all the guidance and help provided during the realization of this thesis. Also I want to thank Enric Moreu's help because while I was learning deep learning he helped me having a well organized project using the newest tools.

Revision history and approval record

Revision	Date	Purpose
0	02/01/2019	Document creation
1	18/01/2019	Document revision
2	22/01/2019	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Juan José Nieto Salas	juanjo.3ns@gmail.com
Kevin McGuiness	kevin.mcguinness@insight-centre.org
Xavier Giró i Nieto	xavier.giro@upc.edu
Eva Mohedano	eva.mohedano@insight-centre.org

Written by:		**Reviewed and approved by:**	
Date	13/01/2019	Date	24/01/2019
Name	Juan José Nieto	Name	Eva Mohedano
Position	Project Author	Position	Project Supervisor

Table of contents

List of figures

List of tables

1. Introduction

Visual attention enables humans to quickly analyze complex scenes by giving to the salient regions higher levels of processing and visual awareness. In Computer Vision (CV) this behaviour is modeled as saliency prediction, it consists of predicting human eye fixations, and highlights regions of interest for human observers. Saliency is represented as heat maps where a higher value of the pixel means more probability to attract human's attention. Saliency estimation is a relevant field because it provides insight into the human visual system, and improves other CV related tasks such as object recognition (Islam et al. 2018) [25] or image retargeting (Dong et al. 2014) [26].

Convolutional Neural Networks are being used by the vast majority of applications in deep learning for computer vision. CNN are a set of layers that with basic operations identify features from images and videos. Current models for video analysis are based on 3D convolutions or LSTMs that are very expensive to train in terms of computational power, therefore we decided to try other approaches such as extend the model input with more information.

Video saliency prediction is still an open problem that has not been studied as much as image saliency prediction, due to its computational power requirements and complex architectures. We can check this statement visiting three of the most recognize leaderboards in the saliency community: Mit Saliency Benchmark[1], LSUN Leaderboard[2] and DHF1K video saliency benchmark[3]. The first two evaluate the models in static datasets and they have over 80 submissions. In contrast, in the third benchmark they evaluate over a video dataset (DHF1K), and there are only 24 submissions. Besides, among the better ten there are only four that are based on dynamic models. Nonetheless, image saliency prediction is the base for video saliency prediction, therefore we have to bear in mind techniques or datasets such as SALICON (Jiang et al. 2015) [6] or iSUN (Xu et al. 2015) [7] that can be useful during the research and development process of a dynamic model.

1.1. Objectives

The main goal of this thesis is to expand a saliency model to the video domain.

To accomplish this objective, we will target in the recently proposed DHF1K (Wang et al. 2018) [2] as a benchmark to train and evaluate the architectures. The main steps to achieve the desired result are:

1. Understand what saliency model is and how do they work. Study ACLNet (Wang et al. 2018) [2], which is the state-of-the-art on video saliency prediction.
2. Set a baseline model based on SalGAN (Pan et al. 2017) [1] on the DHF1K.
3. Explore complementary modalities to explicitly model time dynamics as an input for SalGAN.

[1] http://saliency.mit.edu/results_mit300.html
[2] http://lsun.cs.princeton.edu/leaderboard/
[3] https://mmcheng.net/videosal/

1.2. Work Plan

Project: Research	WP ref: WP1	
Major constituent: Papers	Sheet n of m	
Short description: Study state-of-the-art models in video saliency prediction.	Planned start date:24/9/2018 Planned end date:19/11/2018 Start event: - End event: -	
Internal task T1: Previous SalGAN Internal task T2: ACLNet Internal task T3: DeepVS Model	New model design	Dates:

Project: Pytorch	WP ref: WP2	
Major constituent: Software	Sheet n of m	
Short description: Master main tool to develop the model.	Planned start date: 2/10/2018 Planned end date: 12/10/2018 Start event:- End event:-	
Internal task T1: Installation and set-up Internal task T2: First NN Internal task T3: Pretrained models	Basic NN	Dates:

Project: Docker	WP ref: WP3	
Major constituent: Software	Sheet n of m	
Short description: Tool to ease the management of the entire project.	Planned start date: 8/10/2018 Planned end date: 16/10/2018 Start event:- End event:-	
Internal task T1: Learn docker Internal task T2: Create image	Image with model	Dates:

Project: SalGAN reimplementation	WP ref: WP4
Major constituent: Software	Sheet n of m
Short description: Achieve or even improve Lasagne based SalGAN performance with SalGAN in pytorch.	Planned start date: 2/10/2018 Planned end date: 25/01/2019 Start event:- End event:-

Internal task T1: Train Internal task T2: Evaluate Internal task T3: Fine-tuning Internal task T4: Validate Internal task T5: Add adversarial to current “SalGAN”	Fine-tuned model	Dates:

Project: SalGAN on DHF1k	WP ref: WP5	
Major constituent: Software	Sheet n of m	
Short description: Train SalGAN on DHF1k.	Planned start date: 2/10/2018 Planned end date: 25/01/2019 Start event:- End event:-	
Internal task T1: Design architecture Internal task T2: Code Internal task T3: Permute techniques Internal task T4: Train Internal task T5: Evaluate Internal task T6: Fine-tuning Internal task T7: Validate	Fine-tuned model	Dates:

Project: SalGAN on LEDOV	WP ref: WP6	
Major constituent: Software	Sheet n of m	
Short description: Train SalGAN on LEDOV.	Planned start date: 2/10/2018 Planned end date: 25/01/2019 Start event:- End event:-	
Internal task T1: Design architecture Internal task T2: Code Internal task T3: Train Internal task T4: Evaluate Internal task T5: Fine-tuning Internal task T6: Validate	Fine-tuned model	Dates:

Table 1: Work Packages

Milestones

WP#	Task#	Short title	Milestone / deliverable	Date (week)
1	3	Model understanding	Model design	15/10/2018
2	3	Master main tool	NN	15/10/2018

4	-	Model weights	Train next models from this new baseline	26/11/2018
4	5	Correct weights and parameters	Fine-tuned model	07/01/2019

Table 2: Milestones

1.3. Gantt Diagram

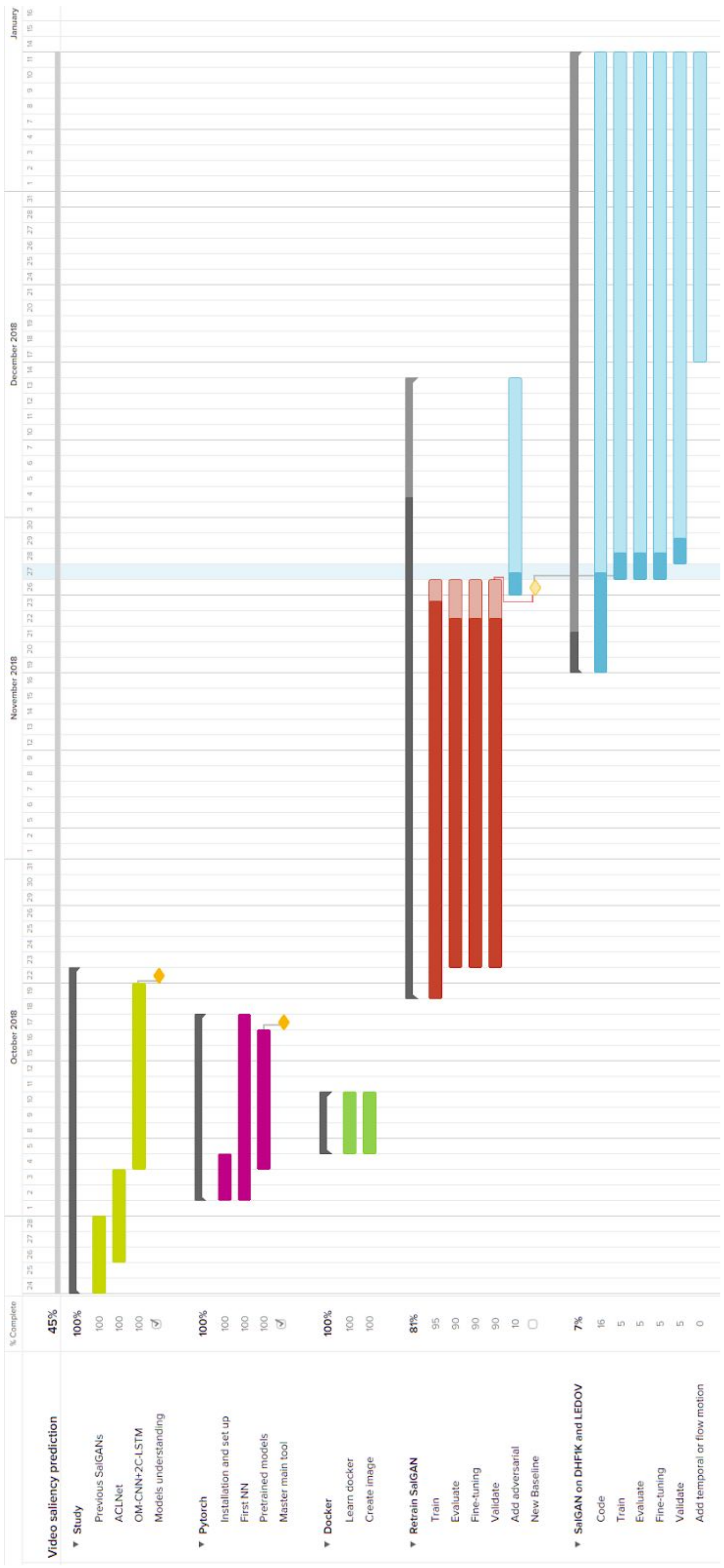


Figure 1: Gantt Diagram

2. State of the art of the technology used or applied in this thesis:

In section 2.1 we are going to introduce some datasets that have been useful to train and evaluate our models. In the next section 2.2 we will study the architectures that have been the base or influence for our final architecture. Finally, in section 2.3 we will review the metrics that allow us comparing the performance of the models.

2.1. Datasets

2.1.1 Introduction

There is no good model without good data in the machine learning world, so during this thesis we have been using the two popular video datasets for saliency prediction, DHF1K and LEDOV (Jiang et al. 2017) [11]. We decided to train our models just using the first one however, we also explored LEDOV dataset, which involved tasks such as learning how to process the ground truth videos and how to evaluate results with the provided matlab scripts[4]. In this section we also introduce the popular SALICON dataset because it was necessary for us to use it to fine-tune the pytorch version of SalGAN, where the ported weights were not matching the performance of the original Lasagne implementation. In this section, we study some aspects from these datasets that can be interesting to know before doing any model training or evaluation. In particular, we are going to focus on the splitting train/validation/test, the scale and quality, the content diversity and some other technical specifications.

2.1.2 SALICON

The name arise from Saliency in Context and uses images from the MS COCO dataset. The saliency fixations were collected using mouse tracking.

Train/validation/test: The dataset includes 20.000 images, where 10k pertain to train, and 5k to both validation and test.

Quality: Each image has a 640x480 resolution.

Content diversity: They sampled 10k images from the MS COCO training set [ref], which contains 80 of the 91 categories. The subset was selected from a total of 17.797 images and the criteria was the number of categories in each image.

Technical specifications: The mouse tracking was made with AMT (Amazon Mechanical Turk) which is a crowdsourcing marketplace that allowed the authors to build the largest saliency image annotated dataset to date.

[4] LEDOV scripts and data are available at github.com/remega/LEDOV-eye-tracking-database

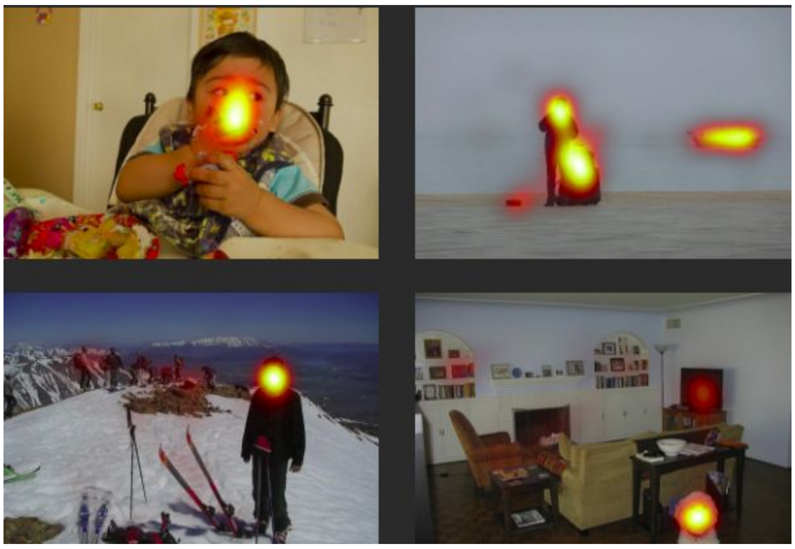


Figure 2: SALICON images with its corresponding ground truth. Source: http://salicon.net/explore/

2.1.3 DHF1K

DHF1K stand from Dynamic Human Fixations 1K it is the largest video saliency dataset.

Train/validation/test: 700 videos out of 1K are for training, the next 100 are for validation and the last 300, for test, are held-out for benchmarking.

Scale and quality: They searched on Youtube around 200 key terms and selected 1.000 video sequences from the retrieval results. All videos were converted to 30 frames per second (fps) and 640x360 spatial resolution, which results of having 582.605 frames with total duration of 19.420 seconds.

Content diversity: They have a total of 150 categories, and also a more generic classification with just 7 categories. Also they have varied motion patterns either of the camera or of the content.

Technical specifications: The fixation points were collected using an eye-tracker called Sensoc Motoric Instruments from 17 participants aged between 20 and 28.

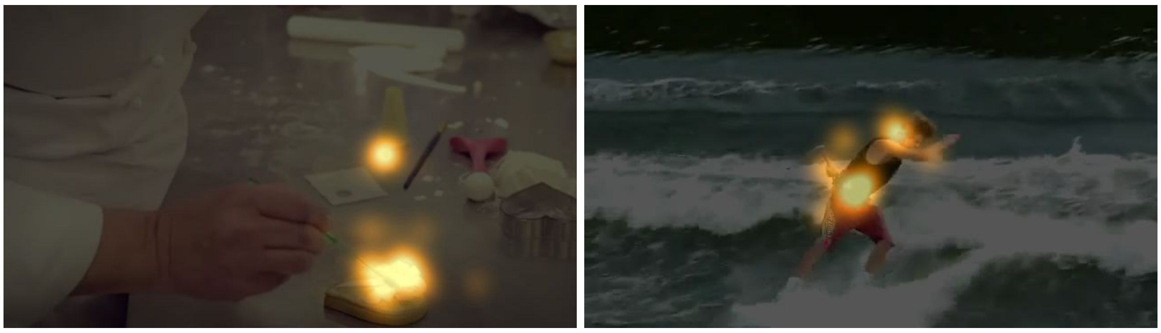


Figure 3: DHF1K images with its corresponding ground truth. Source: DHF1K dataset.

2.1.4 LEDOV

LEDOV stands from Large-scale Eye-tracking Database Of Videos.

Train/validation/test: Their dataset is divided in 436 videos for training: 41 for validation and 41 for test.

Scale and quality: Their videos are at least 720p resolution, which it is the best video resolution in any dataset, with a frame rate of 24 fps giving a total of 6.431 seconds in 538 videos (179.336 images).

Content diversity: The authors took the videos from different online accessible repositories such as daily vlogs, documentaries, movies, sport casts, TV shows, etc. In contrast to DHF1K, they tried to have just stabilize shots or even with any movement by the camera.

Technical specifications: They used a different eye-tracker called Tobii TX300, unlike in DHF1K that it was a binocular, this eye-tracker is integrated on the monitor. Also they have been more generalistic as they had 32 participants aged between 20 and 56.

2.1.4.1 Ground truth:

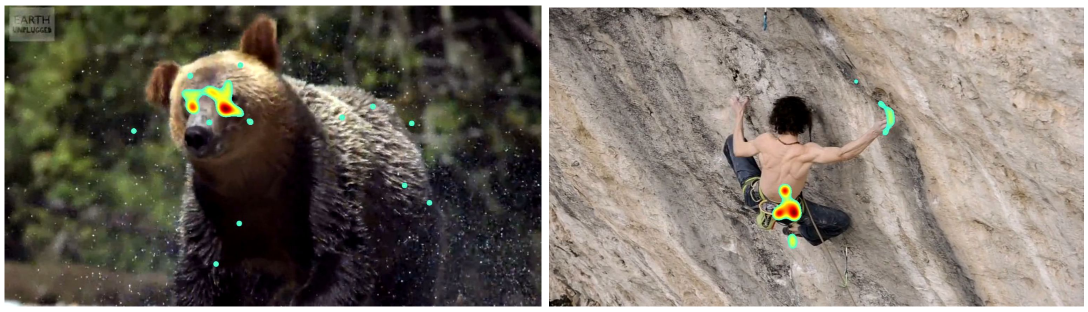


Figure 4: LEDOV images with its corresponding ground truth overlapping. Source: LEDOV dataset.

As you can see in the images from Fig.3 and Fig.4, the ground truth in LEDOV is different compared against DHF1K ground truth, which is more blurred and spread among the salient spot. The ACLNet authors (Wang et al. 2018) did a deeply study of their dataset and they found a few interesting facts. They used YOLO9000 (Redmon and Farhadi 2017) [12] to determine the number of candidates objects of each frame and then compared with the fixations in objects. They saw that as they increase the number of candidates, the fixations points falling into object regions was higher. Iso, the proportion of fixation area inside the total area of the object decreases as the number of object candidates increase. This two statements are represented in this graphs:

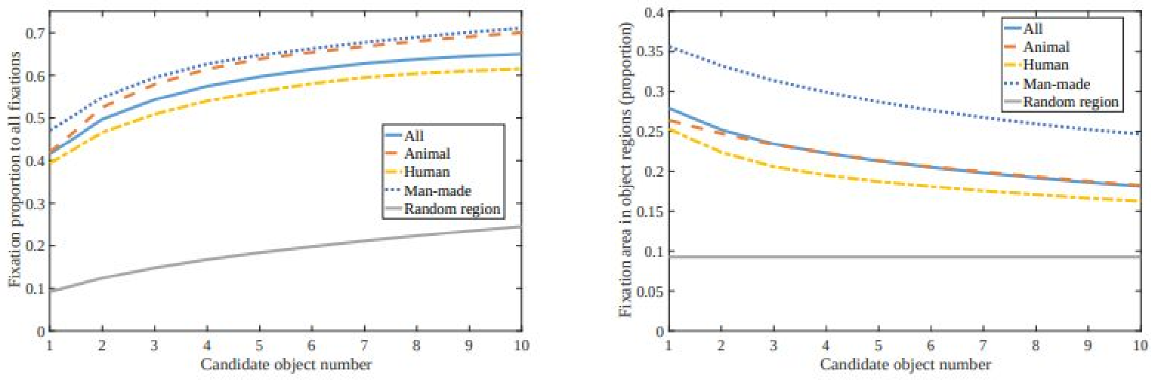


Figure 5: Left: Fixation hit proportion. Right: Fixation area proportion.

This is the reason why the authors use such a big gaussian width mask to generate the ground truth from the fixations points (40 pixels, in SALICON 24 and in DHF1K 30). With this size, it is possible to cover the proportion of objects desired for their dataset.

2.2. Models

2.2.1 Introduction

In this section we are going to analyze different models: the first one, SalGAN (Pan et al. 2017) [1], which is the base of our project, two different models that work with video dynamics (ACLNet (Wang et al. 2018) [2] and DeepVS (Jiang et al. 2017) [11]), and our final model SalBCE.

2.2.2 SalGAN

The SalGAN model is the network on which our work is based. As we will see at a later stage, we made all the modifications and addons from this neural network.

This model was created in 2017 by a group of researchers from the Insight Centre of Data Analytics (where I have been developing my thesis), researchers from UPC and other research groups[5]. It was written in python and lasagne.

Architecture

SalGAN is trained using an adversarial training approach.The architecture consists of a generator and a discriminator. The generator has a convolutional encoder-decoder. On one hand, the encoder consists of a fully convolutional architecture based in the popular VGG-16 (Kim et al. 2016) [8] , using the weights trained on ImageNet (ImageNet: A large-scale hierarchical image database) [10] for image classification. On the other side, the decoder was initialized randomly and with the inverse layers of the encoder, replacing max-pooling with up-sampling layers. The discriminator does the task of distinguishing between ground truth saliency maps and generated saliency maps from the generator.

[5] UPC, Insight Centre of Data Analytics, DCU, Microsoft, Facebook, BSC. (https://github.com/imatge-upc/saliency-salgan-2017)

Consists of pooling layers, fully connected layers with ReLU and Tanh activation functions respectively, except the last one which is a Sigmoid.

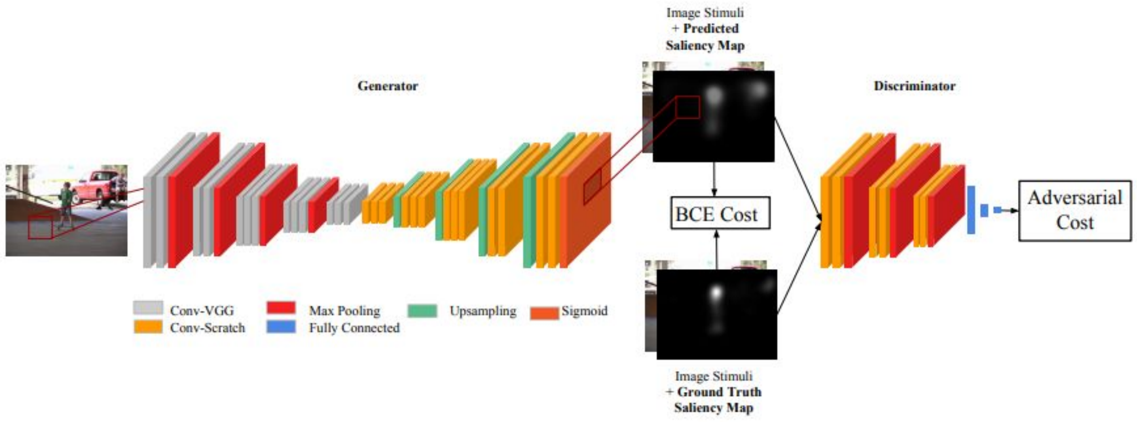


Figure 6: SalGAN architecture: Generator + Discriminator

Loss function

The loss function is a combination between content and adversarial loss.

MSE was used as a baseline reference, they applied an element-wise sigmoid to each output in the final layer so that the pixel-wise predictions can be thought of as probabilities for independent binary random variables. Then, they moved to BCE loss. Given an image I of dimensions N = W×H, the saliency map is represented as a vector of probabilities S, where S_j is the probability of pixel I_j being fixated. A content loss function $\mathcal{L}(S, \hat{S}_j)$ is defined between the predicted saliency map $\hat{S}_j$ and its corresponding ground truth S.

$$\mathcal{L}_{BCE} = -\frac{1}{N}\sum_{j=1}^{N}(S_j log(\hat{S}_j) + (1 - S_j) log(1 - \hat{S}_j))^2 \qquad (1)$$

During adversarial training, the loss function of the saliency prediction network is a combination between the error from the discriminator and the cross entropy with respect to the ground truth in order to improve the stability and convergence rate of the adversarial training.

$$\mathcal{L} = \propto \mathcal{L}_{BCE} + L(D(I, \hat{S}_j), 1) \qquad (2)$$

During the training of the discriminator, no content loss is available and the loss function is:

$$\mathcal{L}_D = L(D(I, S_j), 1) + L(D(I, \hat{S}_j), 0) \qquad (3)$$

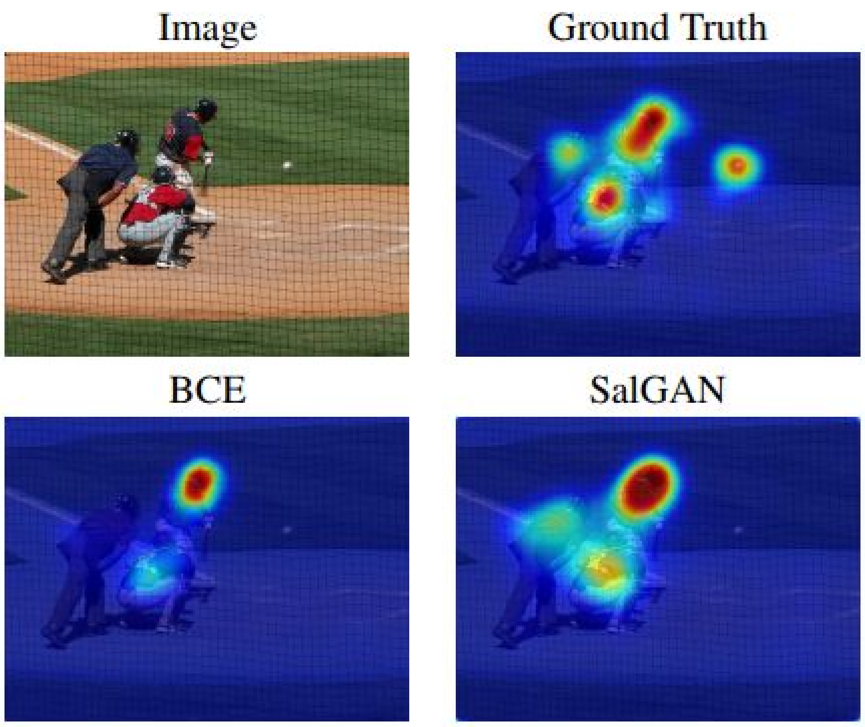


Figure 7: SalGAN comparison among saliency generated with BCE loss and saliency generated with adversarial loss. Source: SalGAN Paper.

2.2.3 ACLNet

In 2017, a group of institutions from China such as Beijing Lab of Intelligent Information Technology or Beijing Institute of Technology among others, launched a whole new environment of human visual system: ACLNet, which is a CNN model designed to predict saliency from videos, along with DHF1K and their new video saliency prediction benchmark. We managed to get a second position in their current benchmark with the static model SalGAN, and our ambition was to overcome their dynamic model with ours.

Architecture

They propose a model that augments the CNN-LSTM network architecture (Donahue et al. 2015) [16] with an attention mechanism (Borji and Itti 2013) [17] to enable fast, end-to-end saliency learning. The attention mechanism explicitly encodes static saliency information, thus allowing LSTM to focus on learning more flexible temporal saliency representations across successive frames. SALICON (Jiang et al. 2015) [6] data is used to train the attention module. DHF1K, HollyWood2 (Marszalek et al. 2009) [15], UCF Sports (Rodriguez et al. 2008) [14] are used to train the entire model he best combination was to use all the datasets together. For their own dataset they used a splitting of (600/100/300).

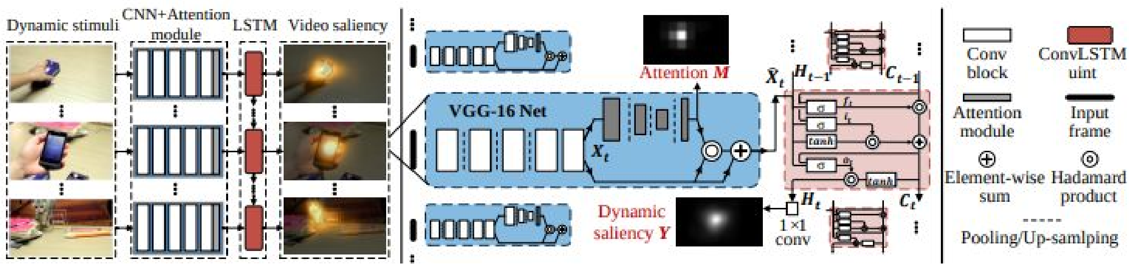


Figure 8: ACLNet architecture composed of VGG-16, Attention module and ConvLSTM.

Loss function

The loss function consists of three different saliency metrics (KL, NSS and CC), because no single metric can capture how good a saliency map is, or in other words, there is no saliency map that can be good in all the metrics at the same time (Kümmerer et al. 2018) [9].

$$\mathcal{L}(Y,P,Q) = \mathcal{L}_{KL}(Y,Q) + \propto_1 \mathcal{L}_{CC}(Y,Q) + \propto_2 \mathcal{L}_{NSS}(Y,P)\,, \qquad (4)$$

where the predicted saliency map is $Y \in [0,1]^{28x28}$, the map of fixation locations is $P \in \{0,1\}^{28x28}$ and the continuous saliency map is $Q \in [0,1]^{28x28}$. $\mathcal{L}_{KL}$, $\mathcal{L}_{CC}$, $\mathcal{L}_{NSS}$ stands for the loss functions of the commonly used metrics to evaluate saliency prediction models. αs are balance parameters empirically set to α1 = α2 = 0.1.

<u>Training protocol:</u> Their model is iteratively trained with sequential fixation and image data. During training, a video training batch is followed by an image training batch. During a video training batch, a loss defined over the final dynamic saliency prediction from LSTM is applied as follows:

$$\mathcal{L}^d = \sum_{t=1}^{T} \mathcal{L}(Y_t^d, P_t^d, Q_t^d)\,, \qquad (5)$$

where (Y_t^d, P_t^d, Q_t^d) denote the dynamic saliency predictions, the dynamic fixation sequence and the continuous ground-truth saliency maps

During an image training batch, they only train the attention module minimizing:

$$\mathcal{L}^s = \sum_{t=1}^{T} \mathcal{L}(M, P^s, Q^s)\,, \qquad (6)$$

where instead of the dynamic maps, the attention map (M) and the ground truth static fixations (Q^s) and continuous maps are considered.

For each video training batch, 20 consecutive frames from the same video are used, both the video and the start frame are randomly selected. For each image training batch, the batch size is set to 20, and the images are randomly sampled from a existing static fixation dataset.

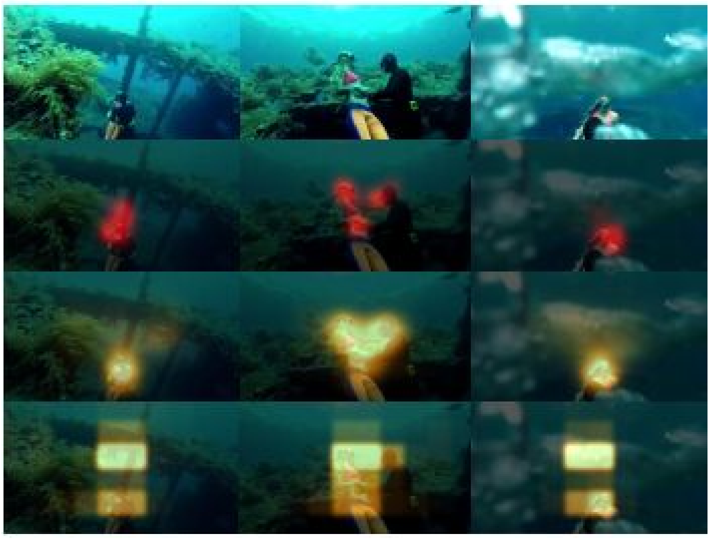


Figure 9: 1st row: Dynamic stimulus. 2nd row: Ground truth. 3rd row: Saliency Prediction. 4th row: Attention Module.

2.2.4 DeepVS

This model was launched with LEDOV in 2017 by Jiang et al. [11] aroused an interest in us because we were thinking of a similar approach for our model which at the end has not been exactly like this.

Architecture

The architecture of this model includes two different streams, the first one named OM-CNN (Object Motion - CNN) integrates two subnets that detect the regions and movement of objects in order to compute the saliency prediction. The first of the two subnets is based on YOLO, the state of the art in video object detection. The second one,checks the movement of these objects using a modified version of FlowNet (Dosovitskiy et al. 2015) [13], a CNN structure to estimate optical flow[6]. Then, the second part of the model is called 2C-LSTM (two-layer convolutional long-short term memory) that uses the saliency maps from OM-CNN as input, to learn dynamic saliency of video clips.

[6] Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. Source: Wikipedia.

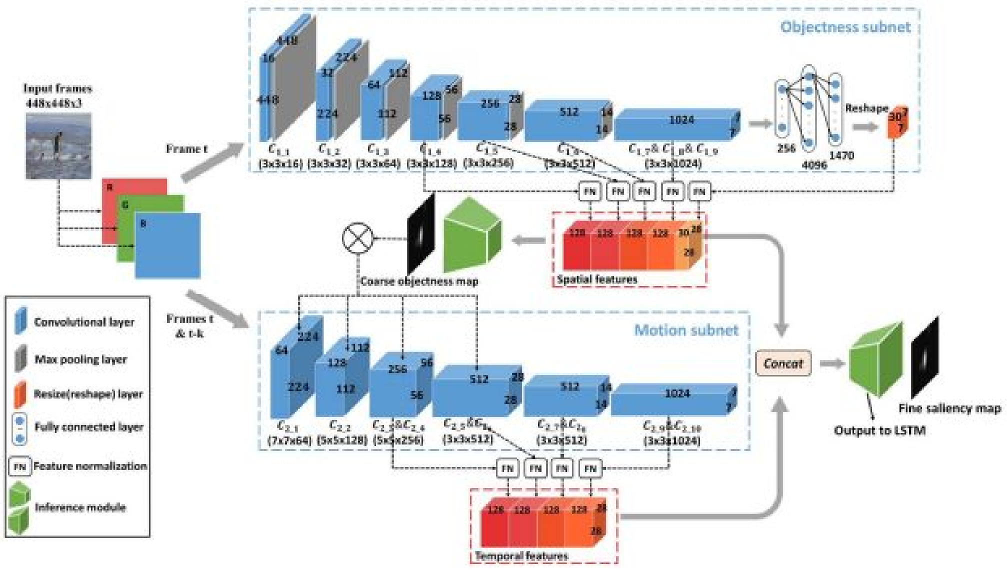


Figure 10: The overall architecture of OM-CNN. Source from [11]

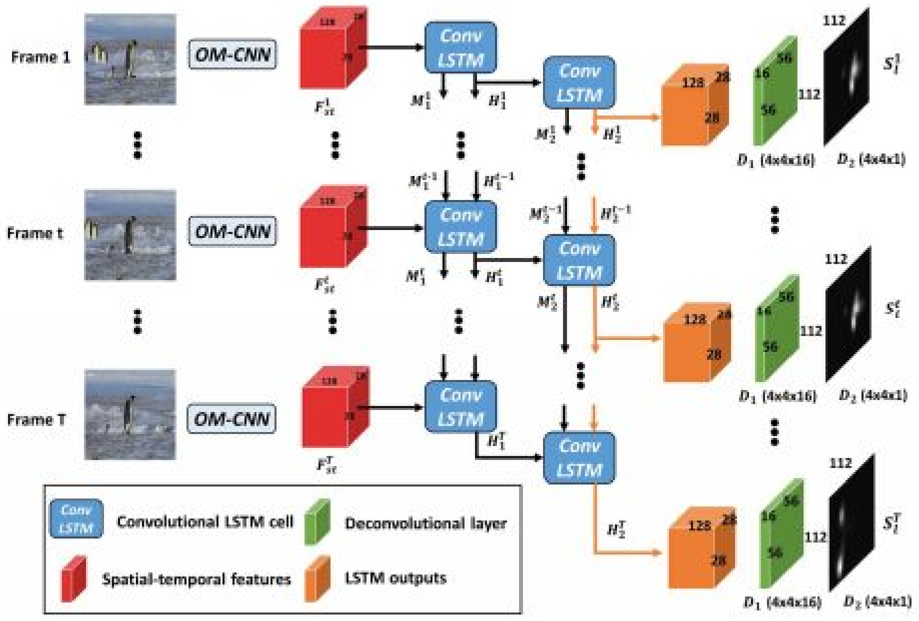


Figure 11: Architecture of 2C-LSTM for predicting saliency transition across inter-frame. Source from [11]

Loss function

It seems that DeepVS group does not agree with the statement of the research paper in saliency metrics where they propose NSS evaluation metric as the metric to rank benchmarks [28]. They use the KL divergence based loss function to update the parameters. In the OM-CNN part of the model they use this loss function:

$$\mathcal{L}_{OM-CNN} = \frac{1}{1+\lambda} D_{KL}(G,S_f) + \frac{1}{1+\lambda} D_{KL}(G,S_c)\,, \qquad (7)$$

where λ is a hyper-parameter to control the weights of two KL divergences and $D_{KL}(G, S_{f/c})$ is the KL divergence between saliency map S_f or coarse objectness map S_c of OM-CNN and ground truth G.

In the second part of the network, the training videos are cut into clips with length T and meanwhile, the parameters of OM-CNN remain fixed to extract spatiotemporal features for each clip. Then, the loss function of 2C-LSTM is defined as the averaged KL divergence over T frames:

$$\mathcal{L}_{2C-LSTM} = \frac{1}{T}\sum_{i=1}^{T} D_{KL}(S_l^i, G_i) \ , \qquad (8)$$

where S_l^i are the final saliency maps generated by 2CLSTM, and G_i are the ground truth of attention maps.

2.2.5 SalBCE

Finally, our model, as we already said is based on SalGAN. But as SalGAN was written in Lasagne[7] (slow learning curve), one of the main concerns at the beginning of the thesis was that SalGAN had to be rewritten in a more popular deep learning framework such as Tensorflow or Pytorch. Therefore, Eva Mohedano ported the generator and the weights to pytorch, though they was not performing as good as the ones from the lasagne version.

Architecture

The proposed model consisted on a simplified version of SalGAN, only considering the encoder-decoder part of the generator. The left part of Fig. 6 describes this network architecture. We just had to rewrite the Dataloader pytorch class to fit our dataset and make some minor changes such as adding batch normalization or adding more channels to the input when required. As we were trying different experiments sometimes we needed dynamic changes both in the model and in the pretrained weights. We finally rewrote the discriminator part in pytorch, but as we did not have the ported weights from the lasagne version, the recommendation of the supervisors was to leave this part on standby because the adversarial training requires a lot of computation and timing, and it is very difficult to make it converge.

Loss function

We used only the content loss of SalGAN, based BCE to compare generated and real saliency maps. We did not train the model using the adversarial loss.

Hyperparameters

During this thesis we tried a couple of optimizers, SGD (Stochastic Gradient Descent) (Ruder 2016) [19] and the Adam optimizer (P. and Ba 2014) [18]. In the initial experiments, only the decoder parameters were tuned, as in the original SalGAN implementation [ref]. In the latest experiments, however, we decided to tune the entire encoder-decoder architecture. Regarding the learning rate, we have been using ReduceLROnPlateau scheduler from pytorch, that reduces the learning rate is the validation loss does not improve after N epocs. In our experiments we set N to 10 epochs

[7] Deep Learning Framework: https://lasagne.readthedocs.io/en/latest/

when we run a single experiment and to 3 when we run a set of experiments. We have considered different learning rate start values between 0,001 and 0,00001. The batch size has been reduced from 32 to 12 due to memory restrictions with the available GPUs, which could have had a negative impact in the final performance of the model.

2.3. Metrics

There are many metrics to evaluate a saliency model and we are going to review 5 of them, the ones that the DHF1K benchmark uses to rank models. But, we have to warn that the way we have been evaluating our model is going to be replaced by a more fair process (Kümmerer et al. 2018) [9]. Currently we evaluate all the metrics in the same saliency map, but it is known that you can not perform well in all the metrics with a single saliency map. The solution is to generate a saliency map suited for each different metric like this:

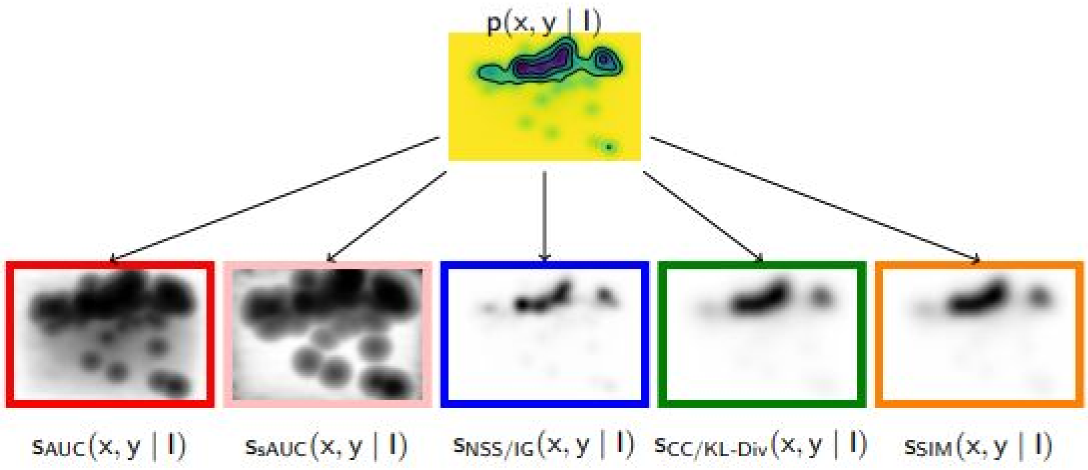


Figure 12: Adapted maps for each saliency metric. Source from [9]

For more details in the implementation of this technique refer to the source of .

Saliency metrics are classified into two main categories: *Location* and *distribution* based metrics.

2.3.1 Location-based metrics

The input ground-truth is represented as discrete fixation locations.

2.3.1.1 AUC-Judd

The Area Under the ROC Curve is one of the most widely used metric for evaluating saliency maps. In particular, de AUC-Judd version, for a given threshold, the true positive rate is the ratio of true positives to the total number of fixations, where true positives are saliency map values above threshold at fixated pixels. And the false positive rate is the ratio of false positives to the total number of saliency map pixels at a given threshold, where false positives are saliency map values above threshold at unfixated pixels.

2.3.1.2 AUC Shuffled

Most of the datasets tend to include a higher density of fixations around the center of the images. So, if a model has a center bias to predict, it will be able to account for at least part of the fixations on an image, independently of the image content. So, sAUC penalizes models that include this bias by sampling negatives from fixation locations from other images, instead of uniformly at random.

2.3.1.3 NSS

The Normalized Scanpath Saliency is the average normalized saliency at fixated locations. It is sensitive to false positives, relative differences in saliency across the image and general monotonic transformations.

2.3.2 Distribution-based metrics

The input ground-truth is represented as continuous fixations map.

2.3.2.1 CC

The Correlation Coefficient (CC) is a statistical method used generally in the sciences for measuring how correlated or dependent two variables are. CC can be used to interpret saliency and fixations maps, as random variables to measure the linear relationship between them.

2.3.2.2 SIM

The similarity metrics also known as histogram intersection, measures the similarity between two distributions, viewed as histograms. It is computed as the sum of the minimum values at each pixel, after normalizing the input maps.

3. Techniques studied / used

In order to adapt a saliency model trained on static images to videos, we considered three main techniques.

The first category consists in modifying the input model to include additional information to the RGB image frames. In particular, we consider adding depth of the scene, and adding a coordinate layer to guide the model to better localize saliency regions. We also consider the dynamic information of the optical flow, however we discard using this in the final experiments given memory space and time restrictions of the project.

The second technique consists in adding a recurrent layer to model the sequential nature of the video data.

Lastly, the third technique consists in applying data augmentation to the video frames.

In this project, we experimented with the first and third techniques, considering the depth information and coordinate layers in the first case. The exploration of recurrent layers and usage of optical flow are left as a future work.

3.1 Modifying the input network

This is motivated by the idea that modeling time dynamics within the model is difficult, so we considered additional information to the RGB as input for the model.

3.1.1 Depth

The depth information has been also considered in this project because the intuition tells us that objects that are closer to the camera tend to be more attended by the observer of a video or image, so they would probably be more salient.

The model we have tried proposed by (Godard et al. 2017) [4], uses unsupervised depth estimation. This model in particular is called ‘unsupervised monocular depth estimation with left-right consistency’.During the model training, instead of using ground truth data of depth maps, the authors use a pair of pictures. Given a calibrated pair of binocular cameras, it is possible to learn a function that is able to reconstruct one image from the other and learn about the 3D shape of the scene that is being depicted.

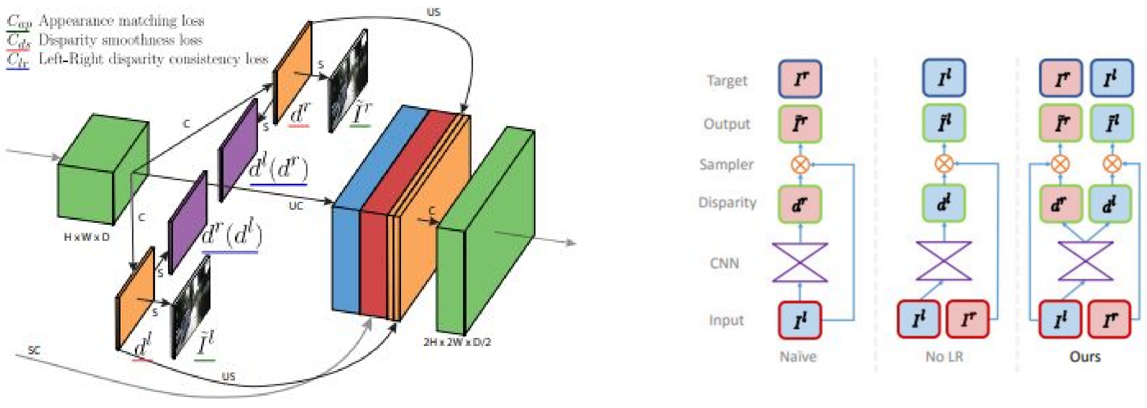


Figure 13: Left and right disparity maps along with the pair of pictures at the output of the loss module.

During the training of the network, two images I^l and I^r, corresponding to the left and right color images captured at the same moment in time are taken. Instead of trying to predict the depth directly, the authors attempt to find the dense correspondence field d^r that, when applied to the left image, enable to reconstruct the right image. d corresponds to the image disparity, which is a scalar value per pixel that the model learns to predict. From here, it is possible to recover the depth map using the baseline distance b between the cameras and the camera focal length f: $\widehat{d} = \frac{bf}{d}$

3.1.2 CoordConv

The idea of this technique, proposed by (Liu et al. 2018) [20], is to provide the convolutional input layers access to its own input coordinates through the use of extra coordinate channels.

The popularity of convolutional layers is due to three main factors: they have relatively few learned parameters, they are fast to compute on modern GPUs, and they learn a function that is translation invariant. According to the paper, CoordConv keeps the two first properties and knows when the third one should be applied. So, with this technique, they say you can improve your model on a diverse set of tasks such as GANs, detection models and reinforcement learning, for example.

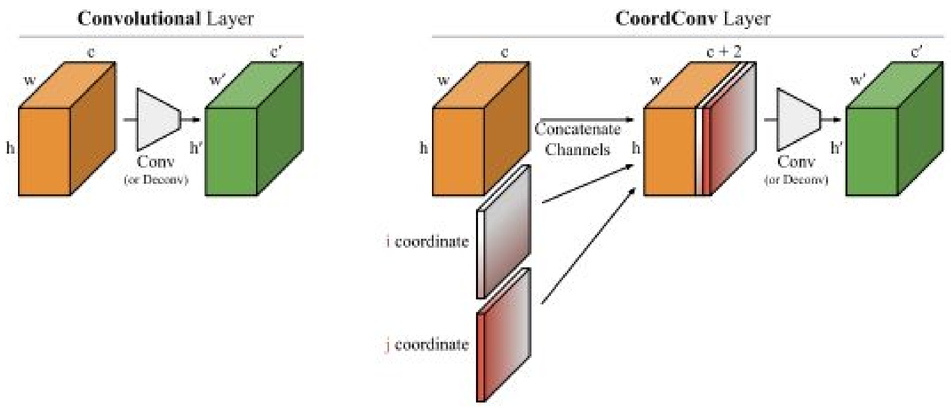


Figure 14: Comparison between usual convolutional layer and coordinates convolutional layer.

3.1.3 Optical Flow

Optical flow is the distribution of the apparent velocities of objects in an image. It is used in computer vision to quantify motion in video streams. We chose to compute optical flow to DHF1K dataset because it is reasonable to think that objects or things that are moving in a scene tend to attract the attention of people, being optical flow a good candidate source of information to develop a video saliency model. We considered two pre-trained CNN models of optical flow: FlowNet2 (Ilg et al. 2017) [13] and SpyNet (Ranjan and Black 2017) [3]. The optical flow information could be used as an additional information of the input image in the SalGAN model. For that, the RGB input channel could be modified to include 2 channels of optical flow. However, given restriction on memory and time, this experiment was discarded and left for future steps. We had to compute the optical flow every 11 frames but from consecutive frames like this:

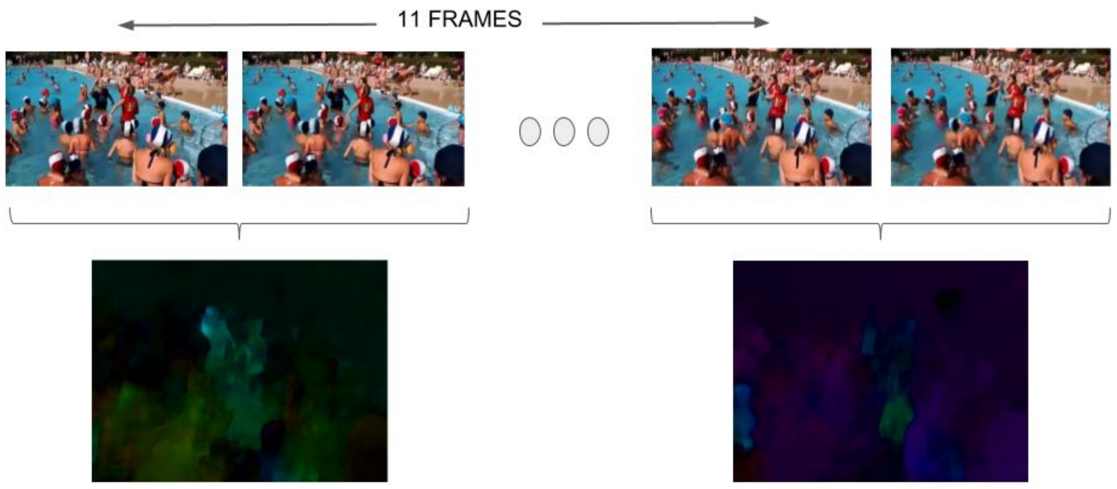


Figure 15: Diagram showing how we computed optical flow. Every eleven frames, two frames are fitted to the SpyNet Network.

Optical flow frames were computed in the DHF1K dataset with SpyNet, because according to the authors, it is better than some of the FlowNet networks despite of being 96% smaller.

The name SpyNet comes from Spatial Pyramid Network. This kind of network can be understood as a recurrent neural network, but instead of time, different resolutions of an image are considered and the optical flow is passed instead of a cell-state. So, as shown in the Figure 16, each network G_k computes a residual flow which is accumulated to, at the end, infer two channels with the velocity of the x and y directions. For each network, the image is increased until the last one where they have the full resolution optical flow of 384x512.

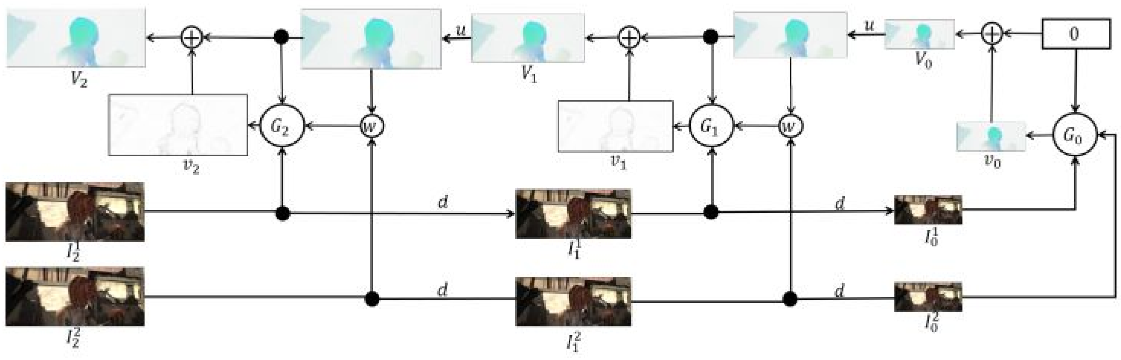


Figure 16: Inference in a 3-Level Pyramid Network. Source from [3]

3.2 Adding recurrent layers

In video saliency prediction, instead of working with individual images, we have a sequences of frames that add temporal context to the saliency prediction. This context is information about the past, information that could tell, for example, that one object disappears from the frame and it is not relevant anymore.

In machine learning, there are many ways of carrying out information from the past to the current state to be able to predict things better. The most well-known are the Recurrent Neural Networks (RNN) (Sherstinsky 2018) [22], Long Short Term Memory (LSTM) (Sherstinsky 2018) [22], Gated Recurrent Unit (GRU) (Chung et al. 2014) [23] or even Hierarchical neural attention encoder (Yan et al. 2018) [29].

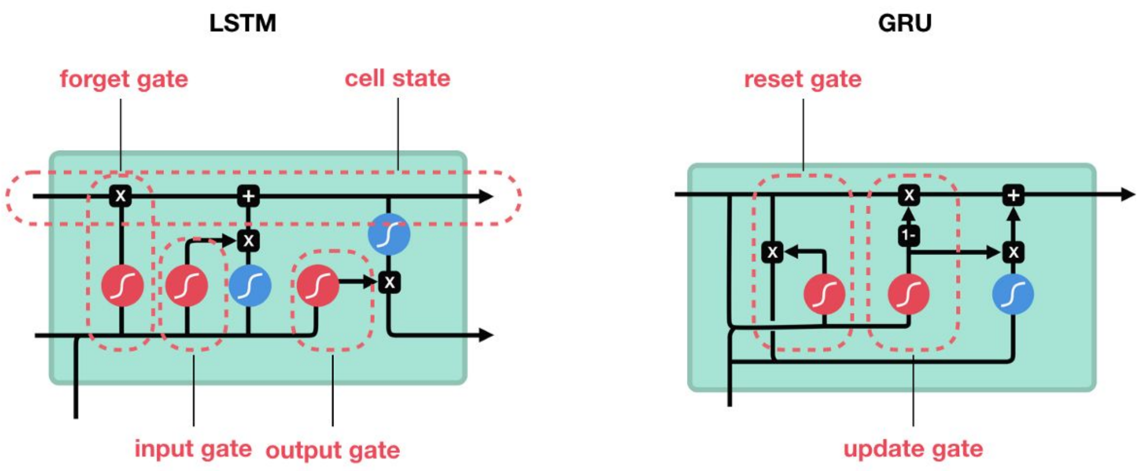


Figure 17: LSTM workflow. Source: [8]

The proposed technique focuses in adding temporal regularization consisted in using a LSTM, in particular a ConvLSTM. The LSTM consist of the following steps:

- First, a forget gate layer indicates what information from the previous state C_{t-1} is going to be kept and what information is going to be removed.
- On the second stage, it is decided what new information is going to be stored on the cell state from a vector of candidates.
- Then, the cell state is updated.
- Finally, the hidden state is updated. For this, the new cell state is scaled between -1 and 1 and multiplied by the output of a Sigmoid function to decide the information that the hidden state preserves.

In the end, we discarded using LSTM to implicity encode the time information within the model and, instead, this research focused in studying alternative ways to include the video dynamics such as optical flow or coordconv.

3.3 Adding data augmentation

Data augmentation is a popular technique used to reduce the network overfitting and to improve the generalization deep learning models. In consists in generating addition training images by, for example, randomly cropping sections of the images, applying

[8] https://cdn-images-1.medium.com/max/1600/1*yBXV9o5q7L_CvY7quJt3WQ.png

random rotations, or/and flipping horizontally/vertically the images. When applied to saliency prediction, a few aspects had to be considered: the ground truth of an image depends on the elements of the image, the shapes, the colors, etc, therefore if you remove some part, the saliency would change and we would not have this new ground truth for the network. For instance, as you can see in the images of the Figure 18, the fact of cropping an image can affect where the person looks or attend within an image. So, as you will see in the experiments section, we decided to apply horizontal and vertical flips and, also, a little rotation to some images and their correspondent ground truth saliency maps.

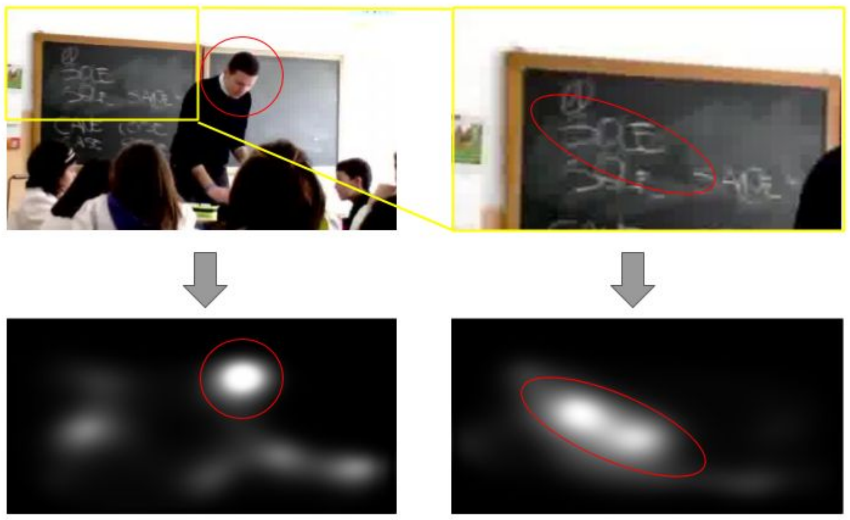


Figure 18: Upper-left: Original frame from DHF1K. Upper-right: Cropped and scaled image (Data Augmentation). Down-left: Original ground-truth corresponding to upper-left image. Down-right: Predicted saliency by our network. As you can see we can not just crop and scale the ground-truth. We don't have the information of how would it be the saliency of that exactly image.

3.4 Adding Batch Normalization

Batch normalization is a technique that enables faster and more stable training of deep neural networks, hence we decided to implement it on our own network. The reasons why this technique has led to better performance in deep learning models has been a very discussed topic. There are some studies that show that it reduces the Internal Covariate Shift (ICS) and as a result it performs better. ICS is a phenomenon where the distribution of inputs to a layer in the network changes due to an update of parameters of the previous layers. But some other studies claim that this correlation is very weak and that there are other properties that benefit the training process such as prevention of exploding or vanishing gradients, robustness to different settings of hyperparameters and keeping most of the activations away from saturation regions of non-linearities. But the most important effect is the reparametrization that makes the gradients smoother, reliable and predictive. (Santurkar et al. 2018)[27]

4. Environment

Given that several configurations were considered, we found highly important to keep a well-structured and organized workflow. This avoids time and memory space usage during the project development. Another important thing is to have everything in different modules. This does not just allow to be more organized but, besides, scales better and makes it easier to rebuild elsewhere.

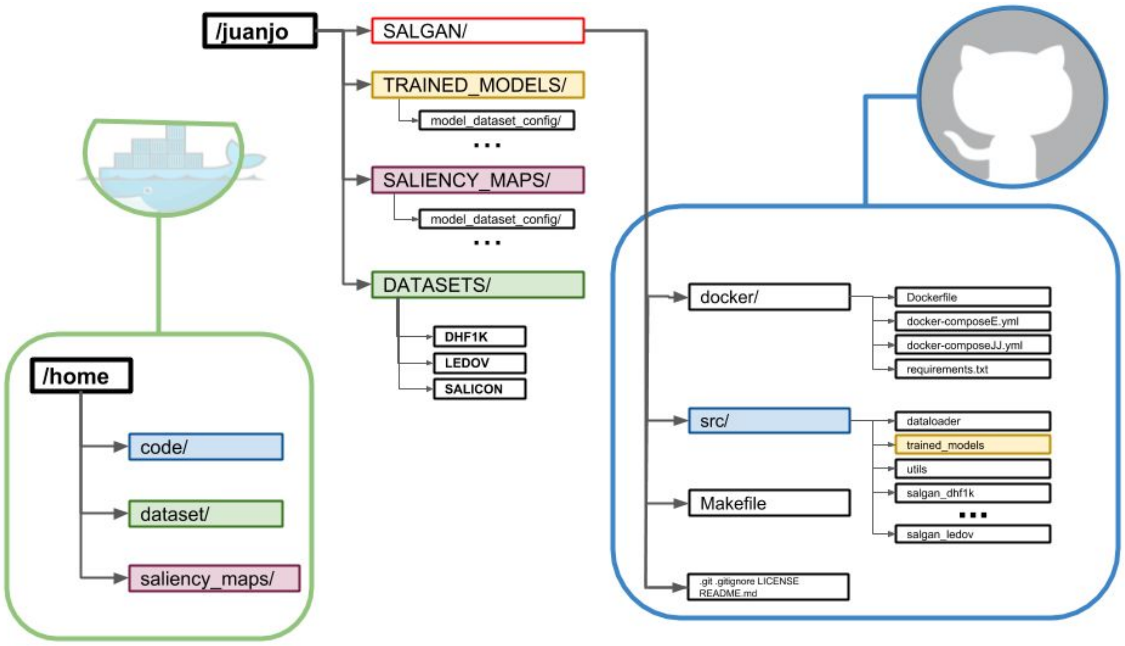


Figure 19: Different modules used: docker container, github repo and the server. Color codes indicates where the folders are mounted into the container.

As shown in Fig. 19, three different modules linked between them were developed:

- Server: this is actually the most fragile part of the architecture, because it relies on us. It can happen that the server could shut-down, the disk could be unmounted or even some folder could be removed because it does not do automatic back-ups. And, of course, we have been struggling with all of these problems. The server stores all the heavy files: trained models, saliency maps generated by the models and the datasets. To be able to interact with this three folders from inside a docker container[9], we have to mount them as volumes. This mount is represented with colors in the image of Fig. 19. There is another main folder in the server that contains a github repo, which is the next module.
- Github repository: to keep our code safe and with controlled versions we use github[10] which includes all the docker files, source code, makefiles and other files. Actually we just mount the src/ folder to the container because we do not need all the rest of files to be inside.
- Docker repository: we created a docker image with just the necessary libraries and instructions to install all the requirements. This image was uploaded to the

[9] https://docs.docker.com/engine/docker-overview/
[10] https://github.com/juanjo3ns/SalBCE

docker hub[11]. We use Makefiles and docker-compose as well to organize and ease the way of launching the containers. As we were using two servers to run experiments, we had to rewrite the different paths of the volumes every time we wanted to run in a different server. So with the two docker-compose it is much easier, and also you can map the ports we want to use later with tensorboard.

With this architecture it is very easy to continue the work in different servers, it is necessary to just pull the github repo and the docker image from their respective websites, and then download the datasets. In this way, it is possible to keep the git and images the most light-weighted possible.

[11] https://cloud.docker.com/u/juanjo3ns/repository/docker/juanjo3ns/salgan_pytorch

5. Experiments

5.1 Checking evaluation

One of the first experiments we did was running inference in either ACLNet and SalGAN, and then, evaluating all the videos available using the metrics defined in Section 2.3. Also, we checked manually the behaviour of the generated saliency maps against ground truth, in order to identify where these models perform better or worse.

A jupyter notebook[12] was developed, allowing the user to conduct an interactive study where it was possible to select the metric to analyse, the model, the higher or the lower metric values, and how many gifs to display. Once selected the desired parameters, a list of different videos with 4 gifs each were generated:

```
Min AUC-JUDD video number:175 According to ACLNet. Metric value: 0.6630850
```

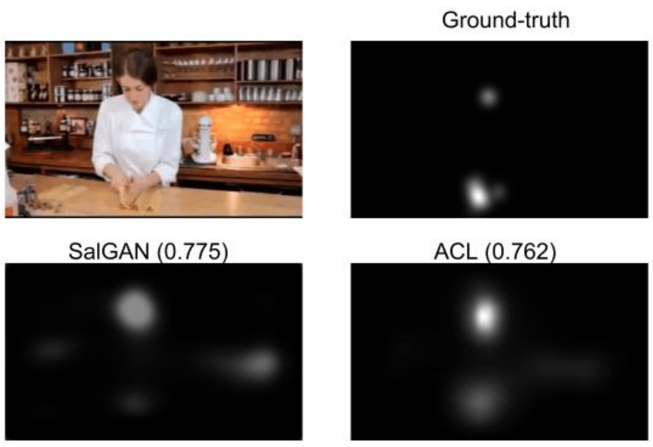


Figure 20: Upper-left: Video number 175 from DHF1K dataset. Upper-right: Ground truth of video 175 provided by DHF1K. Down-left: Saliency generated by SalGAN. Down-right: Saliency generated by ACLNet.

As shown in Fig. 20 we have one of the lowest AUC-Judd with the video number, the value, the ground truth and the saliency map generated by SalGAN and ACLNet. The reason why the model performs bad in this video is pretty obvious: it has a bias on the faces and the ground truth is clearly in the hands and food. If we look to the SalGAN map, we can see that it fails as well. If there are faces on the frames, usually these models will generate saliency parts on them.

So, by developing this notebook, it was possible to identifying some of the strengths and weaknesses of the models. And, in particular, we were interested in what could be improved in the SalGAN model. As we saw, little objects, or objects in movement are not interpreted as something saliencient. In order to solve these particular issues, we could use optical flow or depth, as we already guessed.

[12] Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

Meanwhile we were comparing ACLNet with SalGAN, we submitted the saliency generated over the test set of DHF1K to their benchmark[13]. And we realised that the metrics that we got, even though they ranked number 2 on the scoreboard of all submitted models, they were not as good as we thought. So we started to investigate how we were computing the metrics, and we realised that we were not using the right script neither the suitable maps for some metrics. So we spent some time creating an environment where we could check the metrics easily and we were sure that they were computed right.

One of the comparisons that we did, showed us the distributions of the values of the metrics over the training and validation set:

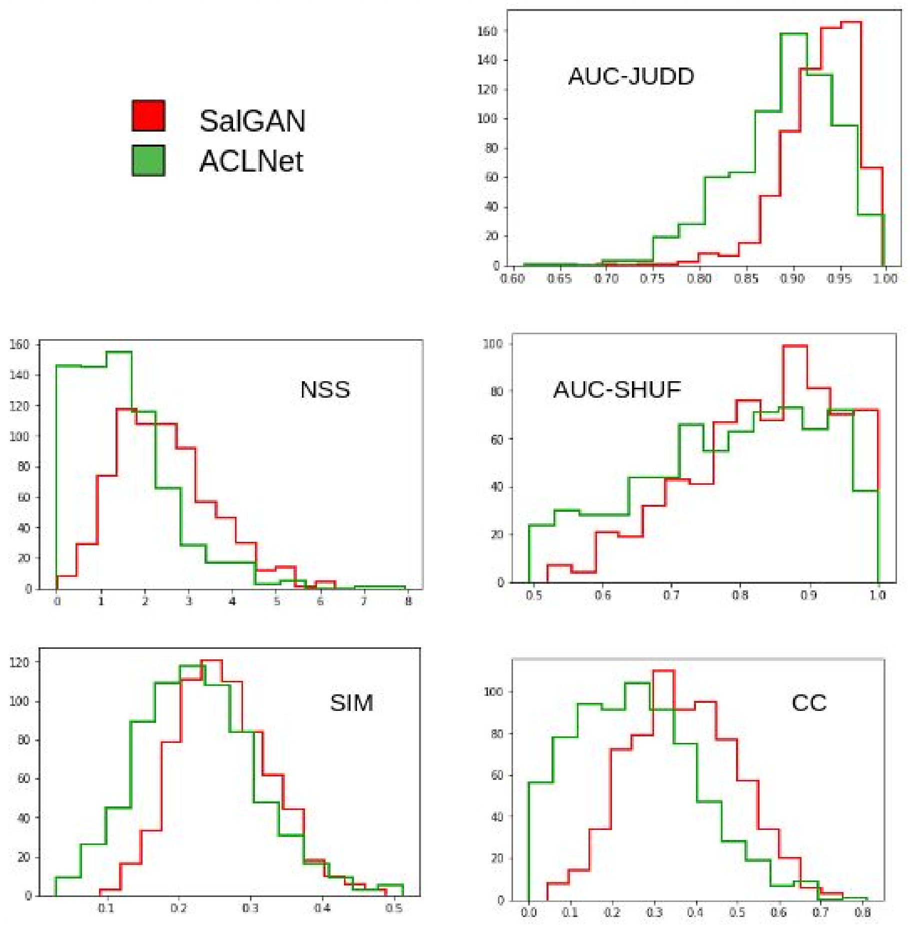


Figure 21: Distributions of the values of the metrics over the training and validation set.

We evaluated metrics using two available scripts [14] [15] in the DHF1k. To check details in these scripts evaluation refer to Appendix 5.

[13] https://mmcheng.net/videosal/
[14] https://github.com/tarunsharma1/saliency_metrics/blob/master/salience_metrics.py
[15] https://github.com/herrlich10/saliency/blob/master/benchmark/metrics.py

5.2 Training phases

5.2.1 Setting baseline model

We used a pytorch implementation of SalGAN for our experiments. However, the ported lasagne weights were not matching the performance of the original model. Therefore, we decided to fine-tune over SALICON the ported weights in pytorch to improve the performance of the new implementation.

VALIDATION	BASELINE LASAGNE	BASELINE PYTORCH	3 EPOCHS SALICON	17 EPOCHS WITH DATA AUGMENTATI ON	27 EPOCHS D.AUGM (+ROTATE) NESTEROV PATIENCE=5
AUC_JUDD	0,856	0,763	0,861	0,862	**0,863**
AUC_SHUF	0,814	0,709	0,832	0,500	0,830
NSS	1,767	1,198	1,757	1,781	**1,789**
CC	0,843	0,555	0,859	0,862	**0,866**
SIM	0,726	0,536	0,751	0,757	**0,761**

Table 3: Evaluation of our model over SALICON dataset in different stages of the training and different configurations.

The optimal model for the pytorch implementation of SalGAN was obtained by fine-tuning the model for 27 epochs in SALICON, applying a data augmentation strategy described in Section 3.3, using Stochastic Gradient Descent (SGD) with Nesterov[16] with a learning rate of 0.001, applying a reduction of 0.1 on the learning rate after 5 epochs if no improvement in the validation loss was achieved. From then on, we used the best model from Table 4. referring to it as the Baseline model in the following subsections.

5.2.2 Transfer Learning to DHF1K

Transfer learning (Salakhutdinov et al. 2011) [24] is applied to adapt SalGAN, trained on SALICON to the DHF1k dataset. With this technique, it is possible to use the weights of a neural network trained for a specific task to solve another related problem.

First, we evaluated how was the performance of the Baseline model in the DHF1K dataset, and then we proceed to improve the model making it more suitable for the DHF1k by transfer learning. For more details on the metrics evaluation on DHF1K consult Appendix 4.

[16] https://deepnotes.io/sgd-momentum-adaptive

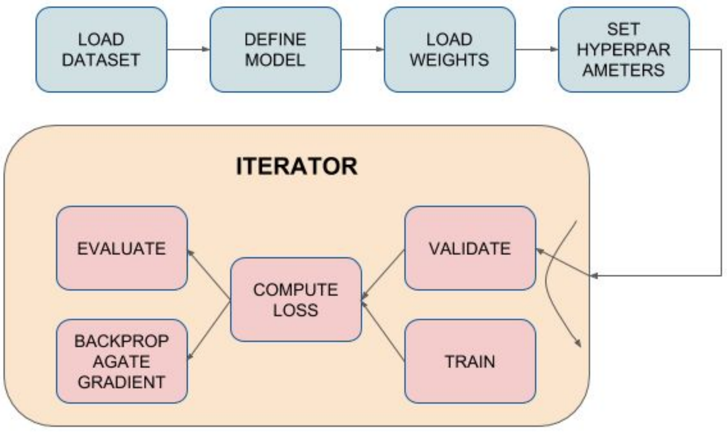


Figure 22: Model workflow.

Figure 22 shows the workflow for this transfer learning experiment. The train and validation Dataloader pytorch class have to be loaded with the correspondent part of the dataset. Then, the model is created with the specified input channels and just after that, loaded with the desired pretrained weights. We have to choose the hyperparameters like learning rate, weight decay, optimizer, batch size, etc. Then there is a loop with the number of epochs we have decided. Inside this loop we alternate evaluation and training. In the first process we just compute the loss and evaluate the validation set. Then in the training process we compute the loss, and backpropagate the gradient to update the unfreezed parameters. Last but not least, we reset the gradients for the next batch size. For more details visit our github repo: https://github.com/juanjo3ns/SalBCE.

Performance of baseline and fine-tuned models is shown in Table 5:

VALIDATION	Baseline	Fine-tuned Baseline
AUC_JUDD	0,872	**0.880**
AUC_SHUF	**0,666**	0.632
NSS	2,035	**2.285**
CC	0,379	**0.420**
SIM	0,267	**0.339**

Table 4: Comparison of Baseline and Fine-tuned Baseline evaluated over DHF1K.

The results indicate that the baseline model greatly benefits from transfer learning by significantly increasing performance in 4 of the 5 evaluated metrics.

5.2.3 Adding extra input signals

In this section, we experiment with additional modalities to add extra information to the network's input. In particular, the experiment was including an extra channel of the depth estimation every 10 frames, this does mean that probably the depth could be even more helpful if we compute it every less frames. The weights for the extra channel in the first convolutional layer were initialized with the average of the RGB weights. For more details in the different experiments used to overcome ACLNet consult Appendix 1. In a similar way, as it is shown in Table 6 with coordconv we managed to improve NSS from the Baseline model, but in general we got a slightly bad performance. The two more channels were initialized with the RGB's weights averages as well.

VALIDATIO N	ACLNET	Train on SALICON	Train on DHF1K		
		Baseline	RGB fine-tuning	RGB and Depth fine-tuning	RGB and Coordconv
AUC_JUDD	0.89	0,872	0.880	**0.895**	0.866
AUC_SHUF	0.601	**0,666**	0.632	0.648	0.629
NSS	2.354	2,035	2.285	**2.524**	2.072
CC	0.434	0,379	0.420	**0.463**	0.389
SIM	0.315	0,267	0.339	**0.351**	0.304

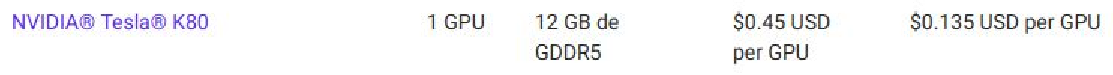
Table 5: Main evaluations over DHF1K including ACLNet metrics and our model progress.

The obtained results provide an evidence that it is possible to enhance the performance of SalGAN to predict saliency on videos by including additional information in the input, which was one of the main ideas of the project. The inclusion of Coordconv, however, does not improve results with respect the baseline. We believe a further investigation is needed in order to properly include this modality. Factors such as normalization of the layer or weight initialization might required further experimentation. The Appendix 2. and Appendix 3 show different combinations of RGB, depth and Coordconv modalities, however best results are shown in Table 6.

5. Budget

All the code, frameworks and tools used to develop this project are open-source, even Matlab (used in some experiments) with the student license has no cost at all. The infrastructure to run all these experiments has been also provided for free by the Insight Centre of Data Analytics, but we could estimate from cloud platforms how much it would cost:

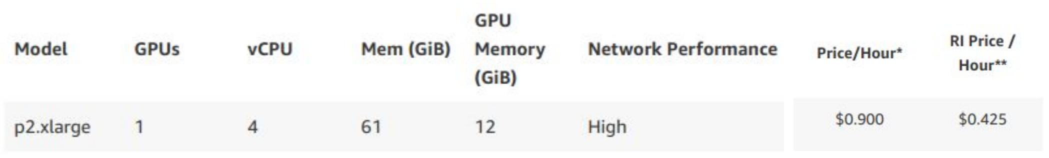
- At GCP (Google Cloud Platform)[17] a similar GPU with the same amount of RAM it would cost 0.135 USD per hour which gives us 340.20 USD for the whole duration. At the end, it would be less because they charge you for the hours you actually use the instance.



NVIDIA® Tesla® K80	1 GPU	12 GB de GDDR5	$0.45 USD per GPU	$0.135 USD per GPU

Figure 23: GCP pricing for similar GPU used during this project.

- At AWS (Amazon Web Services)[18] the same GPU as on Google, it would cost 0.425 USD per hour, almost four times more expensive, giving us 1071 USD for the whole project.



Model	GPUs	vCPU	Mem (GiB)	GPU Memory (GiB)	Network Performance	Price/Hour*	RI Price / Hour**
p2.xlarge	1	4	61	12	High	$0.900	$0.425

Figure 24: AWS pricing for similar GPU used during this project.

But the main cost of this project is the researchers. If we estimate the salary of an undergraduate, a doctorate and 2 senior engineers we have a total cost for the entire semester of 8.700,00€

	AMOUNT	WAGE/HOUR (€/h)	DEDICATION (h/week)	WEEKS	TOTAL
Undergraduate researcher	1	10	35	15	5.250,00 €
Post PhD researcher	1	30	5	15	2.250,00 €
Senior Engineers	2	40	1	15	1.200,00 €
				TOTAL	8.700,00 €

Table 6: Staff involved during the thesis and costs of total hours of work.

[17] https://cloud.google.com/compute/pricing
[18] https://aws.amazon.com/ec2/instance-types/p2/?nc1=h_ls

6. Ethics

Predicting saliency in videos has a slightly controversial dark-side. We are trying to teach computers where they have to look, but the unique way to do that is by showing them how we do it. This could lead to awkward situations since sometimes humans tend to look to sensitive places or spots. We have seen among the datasets we have studied, some bias that could be potentially dangerous in the future. We are not going to show any example because we think it is not necessary to be too explicit in some cases. As we were saying, we have seen in some ground truth data that gazes tend to be in more specific spots when it appears a girl, and the gazes are spreader when a boy appears. This event is accentuated when the activity involving the scene is a sport.

This is just an example, but we could have similar problems with different ethnicities, races or even people from different ages. Even though research groups that create datasets strive to eliminate biases, it is still very difficult to generalize due to scalability problems. Recorded videos take a lot of memory space, so we are quite far from having a video dataset like ImageNet for videos. Hopefully we will have it before having a fully cognitive artificial intelligence fed with a biased model.

7. Conclusions and future development:

During this project, coming from a spread knowledge of deep learning concepts, we achieved a solid idea of how salient deep learning models work, how most famous algorithms are used, the tools necessary to develop and progress through AI projects and even some interesting low level features useful for every kind of computer vision work. With this project we provide the following contributions:

- We have successfully created an evaluation environment for all of our salient models that computes the metrics in the right way and enables you forgetting about lot of steps and lets you focus on your research. The project code is publicly available in https://github.com/juanjo3ns/SalBCE.
- We have studied a state-of-the-art saliency model for static images named SalGAN. We have evaluated its performance in the DHF1K benchmark, rating second in the public leaderboard[19].
- We have successfully implemented a pytorch implementation of SalGAN, porting the weights of the original lasagne model to pytorch, and fine-tuning the weights on SALICON to obtain an equivalent performance to the original implementation.
- Finally, we have investigated how to boost the performance of the baseline pytorch model to predict saliency in videos. The baseline model is fine-tuned on the DHF1k dataset by using RGB information, RGB + Depth, and RGB + coordinates information.

The main goal has been accomplished, having different types of inputs in our network that have improved the saliency metrics over a video dataset. We have seen that adding depth as the 4th channel and adding coordinates as the 4th and 5th channels leads to a model obtaining a better performance than the ACLNet in the validation set, which is the current state-of-the-art model in this benchmark. One of the limitations of our approach is that the studied techniques only consider the previous frame, so we are losing a lot of information from the past. Networks such as LSTM can potentially be more suitable for this task, at the cost of a larger memory overhead and model complexity.

Future steps would investigate the inclusion of LSTM or Optical Flow information to better model the time dynamics.

[19] https://mmcheng.net/videosal/

Bibliography:

[1] 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. *arXiv preprint arXiv:1701.01081*.

[2] Wang, W., Shen, J., Guo, F., Cheng, M.M. and Borji, A., 2018, January. Revisiting Video Saliency: A Large-scale Benchmark and a New Model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4894-4903).

[3] Ranjan, A. and Black, M.J., 2017, July. Optical Flow Estimation using a Spatial Pyramid Network. In *CVPR* (Vol. 2, No. 4, p. 5).

[4] Godard, C., Mac Aodha, O. and Brostow, G.J., 2017, July. Unsupervised monocular depth estimation with left-right consistency. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 6602-6611). IEEE.

[5] Zhi-Jie, L., 2015, June. Image Classification Method Based on Visual Saliency and Bag of Words Model. In *Intelligent Computation Technology and Automation (ICICTA), 2015 8th International Conference on* (pp. 466-469). IEEE.

[6] Jiang, M., Huang, S., Duan, J. and Zhao, Q., 2015. Salicon: Saliency in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1072-1080).

[7] Xu, P., Ehinger, K.A., Zhang, Y., Finkelstein, A., Kulkarni, S.R. and Xiao, J., 2015. Turkergaze: Crowdsourcing saliency with webcam based eye tracking. *arXiv preprint arXiv:1504.06755*.

[8] Kim, J., Kwon Lee, J. and Mu Lee, K., 2016. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1646-1654).

[9] Kummerer, M., Wallis, T.S. and Bethge, M., 2018. Saliency benchmarking made easy: Separating models, maps and metrics. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 770-787).

[10] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L., 2009, June. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 248-255). Ieee.

[11] Jiang, L., Xu, M. and Wang, Z., 2017. Predicting Video Saliency with Object-to-Motion CNN and Two-layer Convolutional LSTM. *arXiv preprint arXiv:1709.06316*.

[12] Redmon, J. and Farhadi, A., 2017. YOLO9000: better, faster, stronger. *arXiv preprint*.

[13] Fischer, P., Dosovitskiy, A., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., Van der Smagt, P., Cremers, D. and Brox, T., 2015. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*.

[14] Rodriguez, M.D., Ahmed, J. and Shah, M., 2008, June. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (pp. 1-8). IEEE.

[15] Marszalek, M., Laptev, I. and Schmid, C., 2009, June. Actions in context. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 2929-2936). IEEE.

[16] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K. and Darrell, T., 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2625-2634).

[17] Borji, A. and Itti, L., 2013. State-of-the-art in visual attention modeling. *IEEE transactions on pattern analysis and machine intelligence*, *35*(1), pp.185-207.

[18] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[19] Ruder, S., 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[20] Liu, R., Lehman, J., Molino, P., Such, F.P., Frank, E., Sergeev, A. and Yosinski, J., 2018. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems* (pp. 9628-9639).

[21] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A. and Brox, T., 2017, July. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1647-1655). IEEE.

[22] Sherstinsky, A., 2018. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *arXiv preprint arXiv:1808.03314*.

[23] Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[24] Salakhutdinov, R., Torralba, A. and Tenenbaum, J., 2011, June. Learning to share visual appearance for multiclass object detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (pp. 1481-1488). IEEE.

[25] Amirul Islam, M., Kalash, M. and Bruce, N.D., 2018. Revisiting salient object detection: Simultaneous detection, ranking, and subitizing of multiple salient objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7142-7150).

[26] Dong, W., Wu, F., Kong, Y., Mei, X., Lee, T.Y. and Zhang, X., 2016. Image retargeting by texture-aware synthesis. *IEEE Transactions on Visualization & Computer Graphics*, (2), pp.1088-1101.

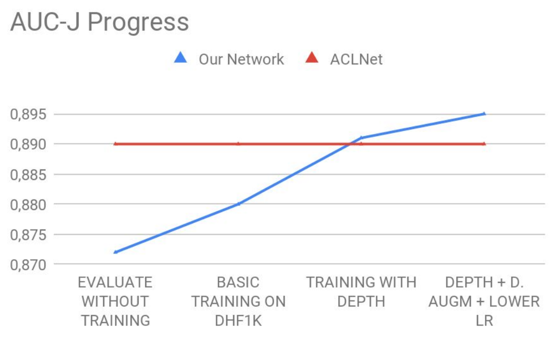
[27] Santurkar, S., Tsipras, D., Ilyas, A. and Madry, A., 2018. How Does Batch Normalization Help Optimization?(No, It Is Not About Internal Covariate Shift). *arXiv preprint arXiv:1805.11604*.

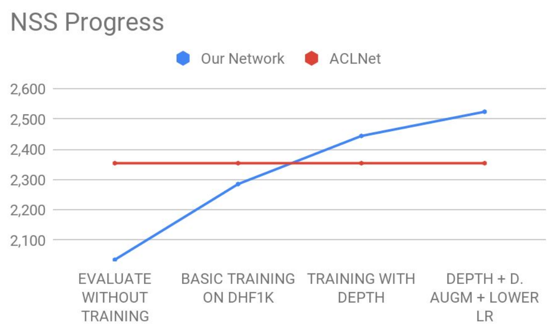
[28] Borji, A., 2018. Saliency prediction in the deep learning era: An empirical investigation. *arXiv preprint arXiv:1810.03716*.

[29] Yan, S., Wu, F., Smith, J.S., Lu, W. and Zhang, B., 2018. Image Captioning Based on a Hierarchical Attention Mechanism and Policy Gradient Optimization. *arXiv preprint arXiv:1811.05253*.

Appendices:

1. Different trainings while overcoming ACLNet





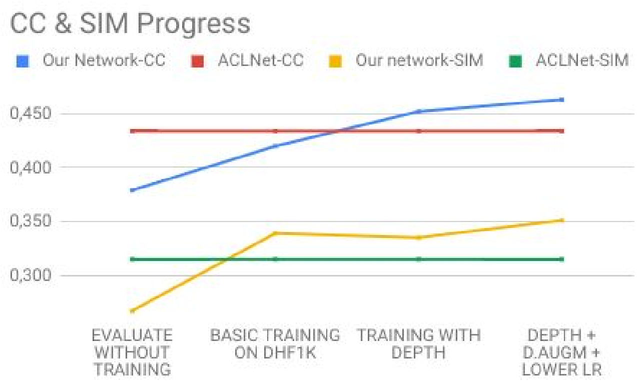
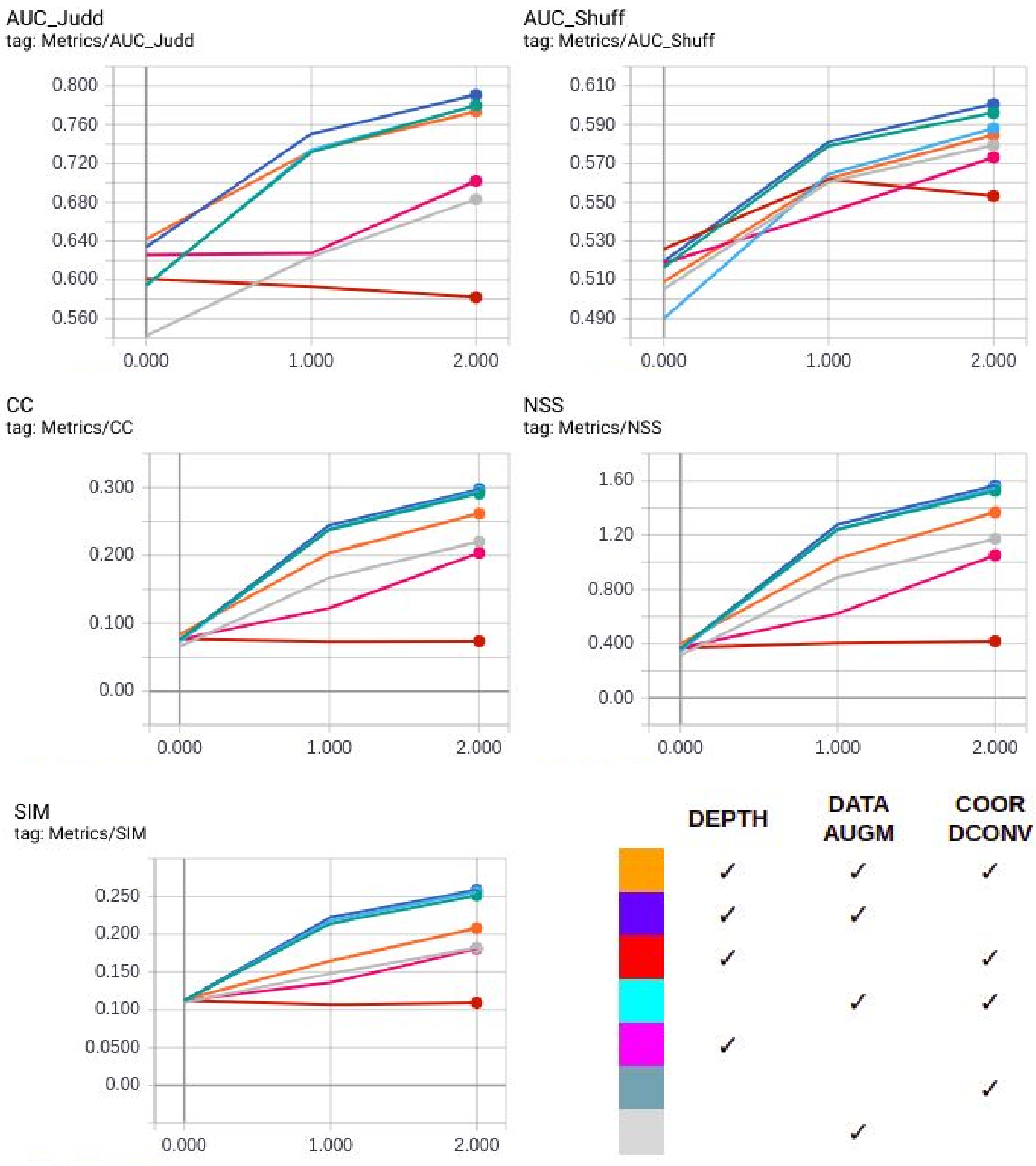


Figure 25: Graphs progress of all the saliency metrics but AUCs which we already beat.

2. Experiments with depth, coordconv and data augmentation

In order to do all the possible combinations among the different techniques, we created a Makefile where we basically permuted them. As we had up to 7 experiments to run, we decided to iterate over just 3 epochs each experiment to see how they perform at the beginning, and then run for more epochs the best one. The first results looked like this:



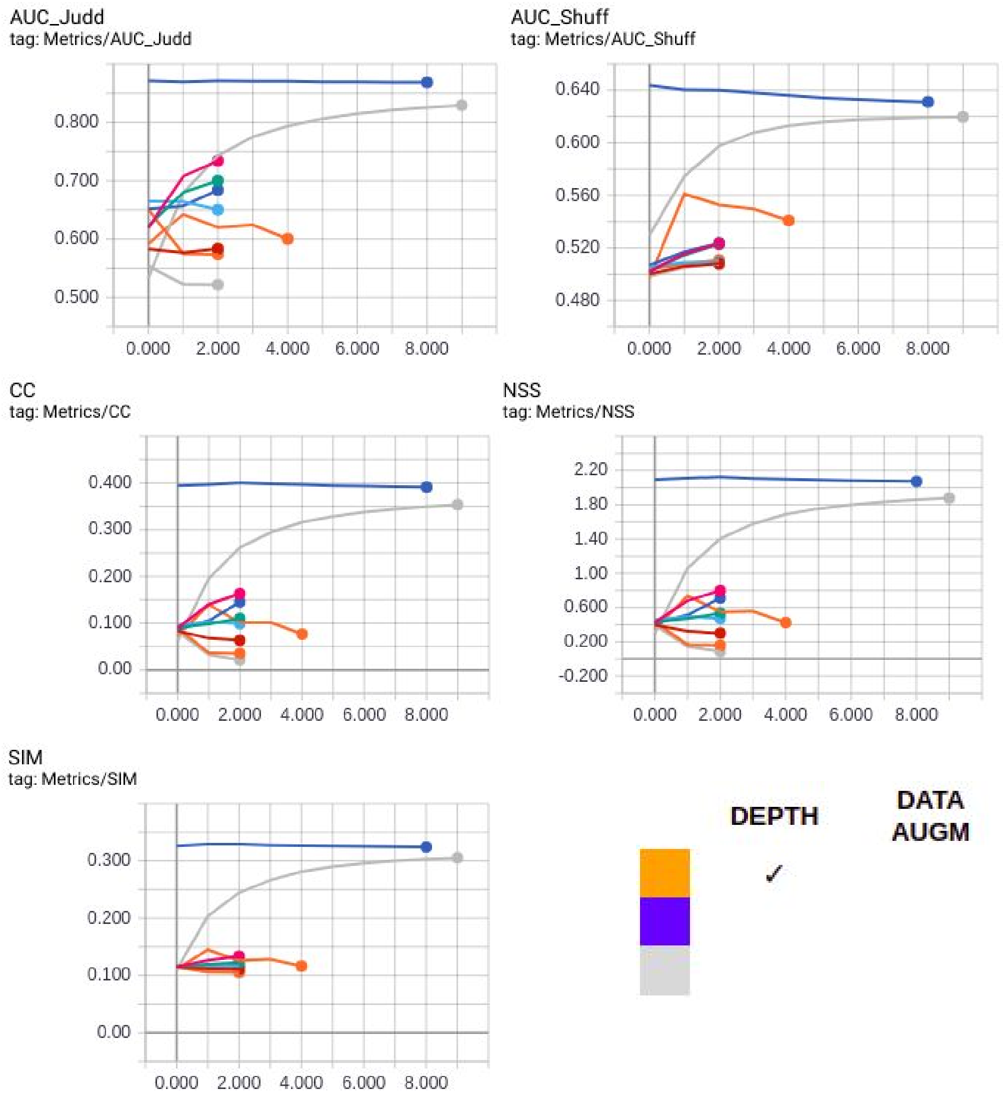
	DEPTH	DATA AUGM	COOR DCONV
🟧	✓	✓	✓
🟪	✓	✓	
🟥	✓		✓
🟦		✓	✓
🟪	✓		
🟦			✓
⬜		✓	

Figure 26: Metric values at each epoch and configuration of each training process.

All the values are slightly below the standards because these metrics are computed during the validation time over 100 frames of every video and then average all together. Usually, when we want the proper metrics, we use another script just for evaluation that loads the weights of the desired experiment and then evaluates over all the frames of the validation set.

The results are very interesting because the best combination uses depth and data augmentation and the second best combination uses just coordconv. But, at the same time, the worst experiments include depth + coordconv and just depth. They do not seem to be consistent with the previous results nor among themselves.

At this point we realised that we were not normalizing between [-1, 1] the coordconv extra channels, which could be a reason of why some previous results were not consistent. So we decided to rerun all the seven experiments we did with depth, data augmentation and coordconv. Along with these experiments, we run another one with depth and coordconv and two more with just coordconv until 10 epochs.



	DEPTH	DATA AUGM	COOR DCONV
🟧	✓		✓
🟪			✓
⬜			✓

Figure 27: Last experiments with coordconv and depth.

In this case, as you can see in Fig. 27, out of the seven 3 epochs experiments, the only one that stands out is the pink. This experiment was trained with the coordinate extra channels. Considering that now coordconv was giving the right saliency spatial references, we decided to train until epoch 10 (gray experiment) which gave us very good results, almost as good as our best model. Our last try (blue experiment), was to fine-tune from the last experiment to overcome our own, but as you can see in Fig. 24 all the metrics remain almost flat, so this does mean that we arrived to the coordconv limit and we'll have to try different things in a future work.

3. Experiments comparing batch sizes

After running the first seven experiments without any improvement, we decided to compare the same experiment with different batch sizes. Both experiments included depth and data augmentation for 6 epochs and a starting learning rate of $1e^{-5}$, patience of 2 and the Adam optimizer. But the second experiment, instead of 12, had a batch size of 32. Until that stage, we had not been able to run experiments with that batch size due to memory space issues in the GPU. But we decided to sacrifice resolution in the network to run this experiment. To do so, we implemented a stride 4 times bigger than the previous one in the first convolutional layer and then the inverse in a upsampler. As it can be seen in Fig. 28, the metrics were still better with batch size of 12, but anyways were worse than our best model.

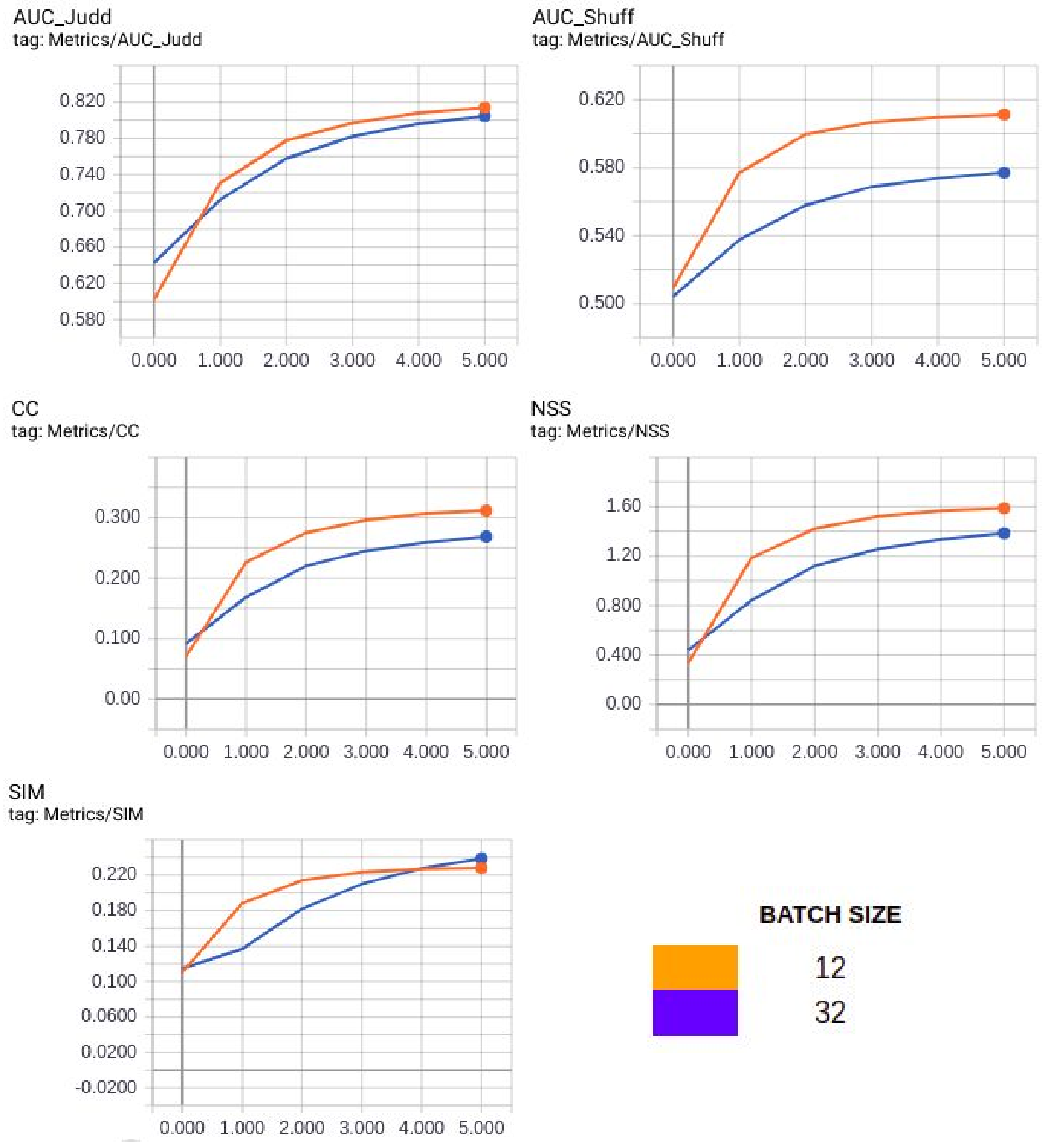


Figure 28: Batch size effect in saliency metrics

4. Differences in metrics evaluation

We observe that the metrics that are computed with discret fixation points (AUCJ, AUCs, NSS) are consistently good in SALICON as in DHF1K. But the metrics that are computed with the continuous saliency map (CC and SIM) are achieving a low performance in the DHF1K dataset. The reason of this difference is because both datasets use the same ground truth fixation maps (just white discrete pixels), but instead, the ground truth continuous saliency maps are different. The network knows a unique way of computing the predicted saliency map, therefore it will evaluate the same predictions against different kinds of continuous ground truth. This leads to higher differences in metrics that use continuous saliency maps.

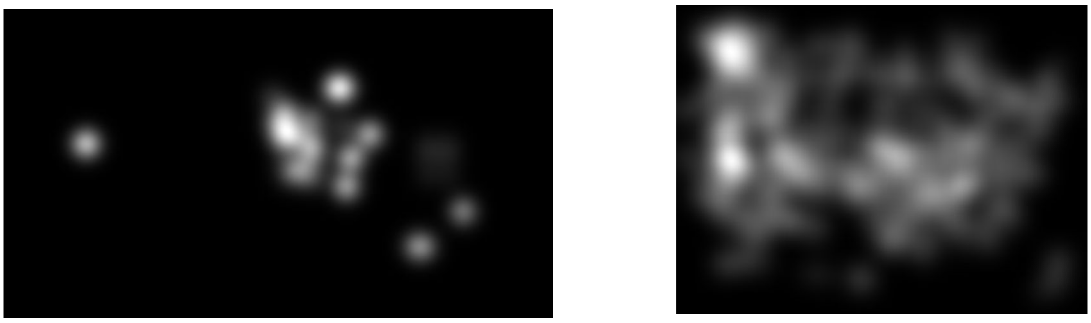


Figure 29: Left: DHF1K ground truth continuous map. Right: SALICON ground truth continuous map.

5. Checking the right script

TEST	ACLNet BENCHMARK	SalGAN BENCHMARK
AUC_JUDD	0,89	0,866
AUC_SHUF	0,601	0,709
NSS	2,354	2,043
CC	0,434	0,370
SIM	0,315	0,262

VALIDATION	SCRIPT 1			SCRIPT 2		
	ACLNet	SalGAN		ACLNet	SalGAN	
		SCALING	NO SCALING		SCALING	NO SCALING
AUC_JUDD	0,894	0,864	0,863	0,890	0,857	0,856
AUC_SHUF	0,849	0,816	0,817	0,610	0,659	0,659
NSS	2,352	1,927	1,927	2,352	1,927	1,927
CC	0,436	0,354	0,354	0,436	0,354	0,354
SIM	0,325	0,256	0,256	0,325	0,256	0,256

VALIDATION	ORIGINAL SCRIPT			MIT SCRIPT		
	ACLNet	SalGAN		ACLNet	SalGAN	
		SCALING	NO SCALING		SCALING	NO SCALING
AUC_JUDD	0,941	-	0,926	0,873	0,848	0,848
AUC_SHUF	0,868	-	0,832	0,616	0,676	0,676
NSS	2,888	-	2,321	2,071	1,834	1,834
CC	0,435	-	0,353	0,379	0,330	0,330
SIM	0,324	-	0,255	0,288	0,238	0,238

Table 7: 1st row: Evaluation of ACLNet and SalGAN over DHF1K TEST set. 2nd and 2rd row: Evaluation of ACLNet and SalGAN over DHF1K VALIDATION set with different scripts with and without scaling.

SalGAN saliency maps are generated with the original lasagne implementation. We also investigate running the evaluation with scaled and not scaled saliency maps (the scaled maps are set to have max and min values of 1 and 0, while in the unscaled maps, max and min values can be smaller and bigger than 1 and 0 respectively). The metrics that are closer to the benchmark metrics are from the script 2. They are not the exactly the same because the benchmark metrics are computed over the test set and our metrics are computed over validation set. Therefore from then on we have been computing the metrics on that script and with resizing to do it faster.