Told you where to look - a multimodal approach for saliency prediction utilizing image captions

Fynn Aurand
Institute of Cognitive Science
University of Osnabrueck
Osnabrueck, Germany
faurand@uni-osnabrueck.de

Elisa Hagensieker
Institute of Cognitive Science
University of Osnabrueck
Osnabrueck, Germany
ehagensieker@uni-osnabrueck.de

Clara Hardes
Institute of Cognitive Science
University of Osnabrueck
Osnabrueck, Germany
chardes@uni.osnabrueck.de

***Abstract*—Machine learning models for saliency estimations have various applications, such as human-robot interaction, virtual reality or image compression. Multimodality in machine learning aims to integrate multiple modalities and to solve its unique challenges. This paper explores the relationship between language and visual perception by investigating the impace of image captions on predicting saliency maps. Two autoencoder models are trained, with the first model trained on images only and the second model fed with additional image caption embeddings to the decoder. We hypothesize that adding image captions can improve the accuracy of our model. However, the results do not support our hypothesis and show that both models perform similarly well. But we did show that using the pretrained weights of our model to generalize on a second dataset improves the performance compared to not using pretrained weights.**

***Index Terms*—AI, Autoencoder, BERT, Convolutional Neural Network, Embeddings, Feature Representation, Fixations, Gaze Tracking, Language model, Machine Learning, MS COCO dataset, Multimodality, SALICON dataset, Saliency map, Supervised learning, UNet, VGG-16**

I. Introduction

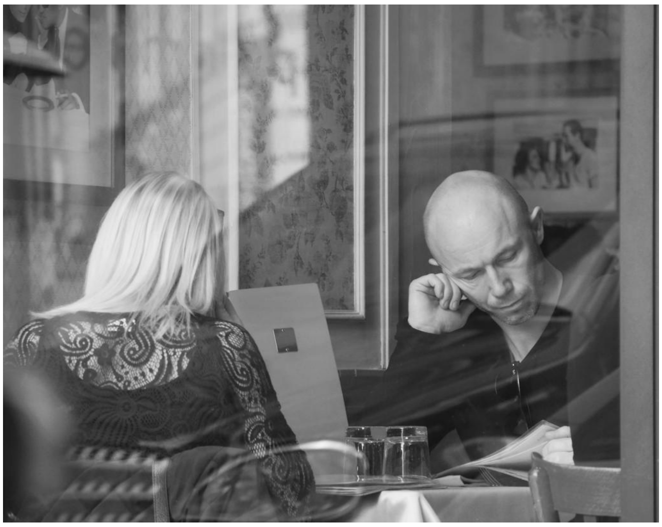


Fig. 1. Self-test: Where are you looking at?

What did your attention focus on first when you looked at the picture 1? Most people would probably look first at the face facing us and the person sitting with his back to us. If we only look at the picture briefly, that's probably all we focus our attention on before we need to describe the picture in more detail.

This paper reports on a two-week project in the ”Implementing ANNs with Tensorflow” seminar in which we explored this topic, the focus of attention. In a group of 3, we investigated how performance in predicting saliency maps changes when captions are provided as input in addition to images. Below we will walk you through our project, starting with what we are doing, why we chose the topic, and what other work already exists. Then we will focus on the methods we use, explain the architecture of our model, and finally present our results. At the end, we will have some ideas on what could be improved and studied in the future.

A. Motivation

We chose this topic because we are interested in exploring the relationship between language and visual perception. Specifically, we wanted to investigate whether providing natural language descriptions of images could improve the accuracy of saliency map predictions. To understand what research has already been done in this area, we conducted a literature review and found some studies that have explored similar topics. One study found that incorporating information about attention can improve generating image captions. However, we did not find any studies that directly investigated the effect of image captions on saliency map prediction performance. As humans, we would benefit from obtaining additional textual information about an image to predict where others may focus their attention. Similarly, we would like to investigate whether AI models can also benefit from such information.

B. Our Project

The field of computer vision, a subset of artificial intelligence, enables machines to interpret, analyze, and understand visual data from the real world. In our study, we will use computer vision to predict saliency maps that are useful in a variety of areas, such as human-robot interaction (HRI), virtual reality (VR), advertising, and image compression. The aim is to train two models, that differ in their input to the decoder and predict where people will focus their visual attention when presented with an image. This is represented by a saliency map that highlights the pixels of the image that are most likely to attract attention.

Hypothesis: We expect that image captions can improve the accuracy of predicting saliency maps. Further, we hope to find our that our model generalizes well when being evaluated on a different dataset with the same modalities.

To test this hypothesis, we will compare the performance of our Autoencoder, which is trained on images only, with an Autoencoder that is fed with image caption embeddings to the decoder. We will use the SALICON dataset [6] and a subset of MS COCO [9] that provides captions for the multimodal input. We can directly compare the performance of both models, because the model has the same architecture, loss and optimizer and we use the same dataset with one more modality used for the second model.

Due to limited computing resources and a relatively small dataset, the primary focus of our project is not on achieving the best performance, but on determining to what extent image descriptions can improve the performance. We will use a model similar to SimpleNet [13], which is a simple model that predicts saliency maps and is therefore suitable for our project idea. However, since SimpleNet is implemented in PyTorch, we will port this model in a similar way to TensorFlow. SimpleNet generally operates only on images as input, so we will modify the model to handle multimodal inputs. To achieve this, we will use BERT [3], a language model, to obtain caption embeddings that will be concatenated with the encoded images and processed in the decoder.

C. Relevance

It is evident that primates need to identify significant regions within a complex scene to orient themselves, work efficiently, and communicate with others.

But why is saliency map prediction relevant to the above domains? In HRI, having a robot that can effectively interact with humans is crucial. This includes having a robot that understands how people perceive their environment and how they communicate [14]. Increasing the performance by adding captions, you achieve better understanding at a higher level of interaction - visually and through speech. Overall, saliency maps show the parts of an environment that are most important to humans, which improves the robots ability to interact naturally.

Saliency maps can also be useful for image compression by preserving the most important parts of an image in better quality [8]. This can be achieved by giving these pixels a higher priority during compression with lossless coding.

Ads can be optimized by ensuring that the most important elements are seen by potential buyers, which can lead to a higher likelihood of action on the part of the buyer.

Virtual Reality and gaming can benefit from the saliency prediction in many ways. Firstly, more engaging gaming experiences can be created when understanding what objects and scenes are most likely to capture the attention. In addition, one can improve the user interface to make it easier for players to access the features. In the future, it might also be used to make customized game experiences if individuals' saliency maps can be reliably predicted.

II. Literature

In the following related work is presented separated into related network architectures, datasets and saliency predictions.

A. SimpleNet - more efficient and interpretable results

Contrary to the common assumption that deeper networks perform better than less complex ones, SimpleNet shows that even a smaller network with less learnable parameters and no unnecessary nonlinearities can achieve comparable or even better performance. Reddy *et al.* [13] compared two networks - MDNSal and SimpleNet. MNDSal consists of two parts, a feature extraction network and a saliency prediction network. The output of the prediction network is a Gaussian Probability Distribution from which a saliency map is predicted. In comparison, SimpleNet is an end-to-end model with a symmetric structure. The number of feature maps increases and decreases symmetrically with depth. A sigmoidal activation layer is the final layer that generates the saliency map. Both networks receive their input from pre-trained architectures that are later fine-tuned to the task at hand, but differ in their readout architecture and loss functions. In our work, we use the SimpleNet architecture. Both architectures achieve comparable performance, with SimpleNet being faster to train and less complex.

Since we want to test the assumption that image descriptions lead to better performance, it is less critical for our project to achieve very high performance and instead use a simple network that can be trained quickly. Additionally to the existing model, it is our aim to integrate embeddings of image captions.

B. Modelling attention

A study in the field of HRI was conducted to train a robot to direct its attention towards objects of human interest with reinforcement learning. The robot was to learn to recognize salient objects and to integrate pointing gestures and spoken cues. It is a follow-up study that is focusing on machine learning instead of psychological features leading to an improvement in the ability to focus the intended target objects [14]. This result is consistent with our hypothesis that multimodal information helps to better predict salient regions.

Xu *et al.* [17] presents a model that generates captions (e.g., for MS COCO) using visual attention. An attention mechanism that receives the current hidden state and a set of image features from the CNN as input is used to compute attention weights that indicate the importance of image regions for generating the next word. These weights are used to compute a weighted sum of image features that is passed as input to a recurrent neural network (RNN) to generate the next word. In this way, the RNN can focus on different parts of the image at each time step using attention. In this work, attention is thus an important component to create accurate and meaningful captions.

DeepGazeIII [7] is a deep learning model to simulate human scanpaths during free-viewinng images. It gets the image together with previous made fixations in the scanpath to predict the next fixation. It does so by using a probabilistic generative model, that outputs a conditional fixation distribution indicating the expected next fixation. DeepGazeIII is pretrained on the SALICON dataset consisting of mouse traces. Contrary to DeepGazeIII, we will not have a temporal dependency in our model, but predict a saliency map that should match the human fixations.

In 2019, a paper was published that creates and uses the Capgaze dataset to implement models for image captioning [5]. Based on the dataset, which consists of images with captions and eye movements recorded synchronously, several hypotheses are proposed, mainly related to human and machine attention during image viewing. However, one of them is of particular relevance to our project. By additionally providing image saliency with soft attention as input, they aim to increase performance in caption generation. To this end, a saliency prediction, perception and language model is used. The saliency prediction model is used to point out salient regions that serve as input to the perception model. This model outputs a feature vector for the perceived region, which finally serves as input to the language model along with the hidden state of the LSTM. They show that the model improves performance when it adds recognition information compared to their soft attention model that highlights all regions in the image.

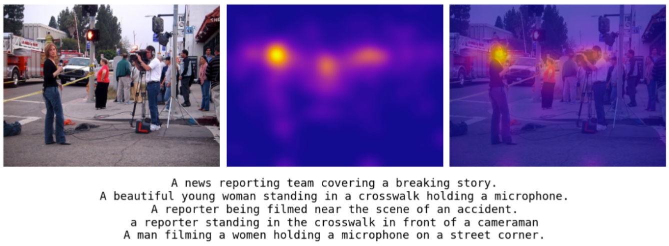
Our project uses a similar dataset to predict saliency maps instead of the image captions. However, the idea is similar: we assume that captions convey prior knowledge about where to look in the image. We use a language model not as the final module, but in between to obtain the embeddings that can be fed into the decoder.

C. Datasets

We use the SALICON dataset [6] which is currently the largest dataset available for saliency prediction. It contains 10000 training images, 5000 validation images and 5000 test images images and is a subset of the MS COCO dataset [9]. MS COCO contains 330.000 images with 5 captions per image, while SALICON provides saliency maps that result from mouse-tracking data of 60 participants. Jiang *et al.* found that mouse tracking closely resembles eye movements and can therefore be used as a new paradigm to measure people's attention. Therefore, the SALICON dataset is widely used in saliency prediction research and many models are first trained on the SALICON dataset and are then finetuned on another dataset.

The capgaze dataset [5] contains 1000 images of everyday situations taken from the Pascal-50S dataset [16]. In addition, for each the eye gaze along with the captions of 5 participants are recorded simultaneously. The dataset was originally used to compare attentional behavior in free viewing and image description tasks. Capgaze would have been a very suitable dataset in terms of the data provided. However, due to the few data samples it would have been too small for training the model. We decided to use this dataset for validating our model on its ability to generalize. Since the provided fixation maps are not from a free-viewing task (as is the case with SALICON), but from an actual labelling task, we believe that the multimodal model should have an advantage here and will perform/generalize better on this dataset, given that the labels we feed into the model will have an even closer relationship to the input images.

D. Data preprocessing



```
A news reporting team covering a breaking story.
A beautiful young woman standing in a crosswalk holding a microphone.
A reporter being filmed near the scene of an accident.
a reporter standing in the crosswalk in front of a cameraman
A man filming a women holding a microphone on a street corner.
```

Fig. 2. Data sample: Original image (left) with its saliency map (middle) and overlaid saliency map (right). The saliency map is to be predicted by the model. One image captions is fed into the model with multimodal input.

III. Methods

Now we will describe the methods we used to address the problem. We will start by providing a more detailed explanation of the dataset and the preprocessing steps. Next, we will visualize the model and explain its construction to solve the problem at hand. It is further detailed in the next section on the implementation. Furthermore, we will explain how the image descriptions were processed to enable the model to handle them effectively. We had to make some adaptations to the model to ensure that it could work with the descriptions properly. The structure of our pipeline for the autoencoder is provided in 3.

The whole code is implemented in Python, while we make use of the following modules in particular: `tensorflow`, `keras`, `pycocotools`and `scipy` and `pandas`. We also load the BERT model from `tensorflow_hub` to retrieve a sentence embedding of the caption, and use a pre-trained implementation of VGG-16 [15] from `tf.keras.applications` as the backbone network in our encoder to obtain the image embeddings.

What is required are images with associated captions and saliency maps, as shown in 2. However, for the SALICON dataset we only have images with fixation annotations or images with caption annotations. Therefore, we collect the matching labels from the COCO [9] dataset to match the images from the SALICON [6] dataset, which we can locate using the image IDs. We then retrieve the fixations and

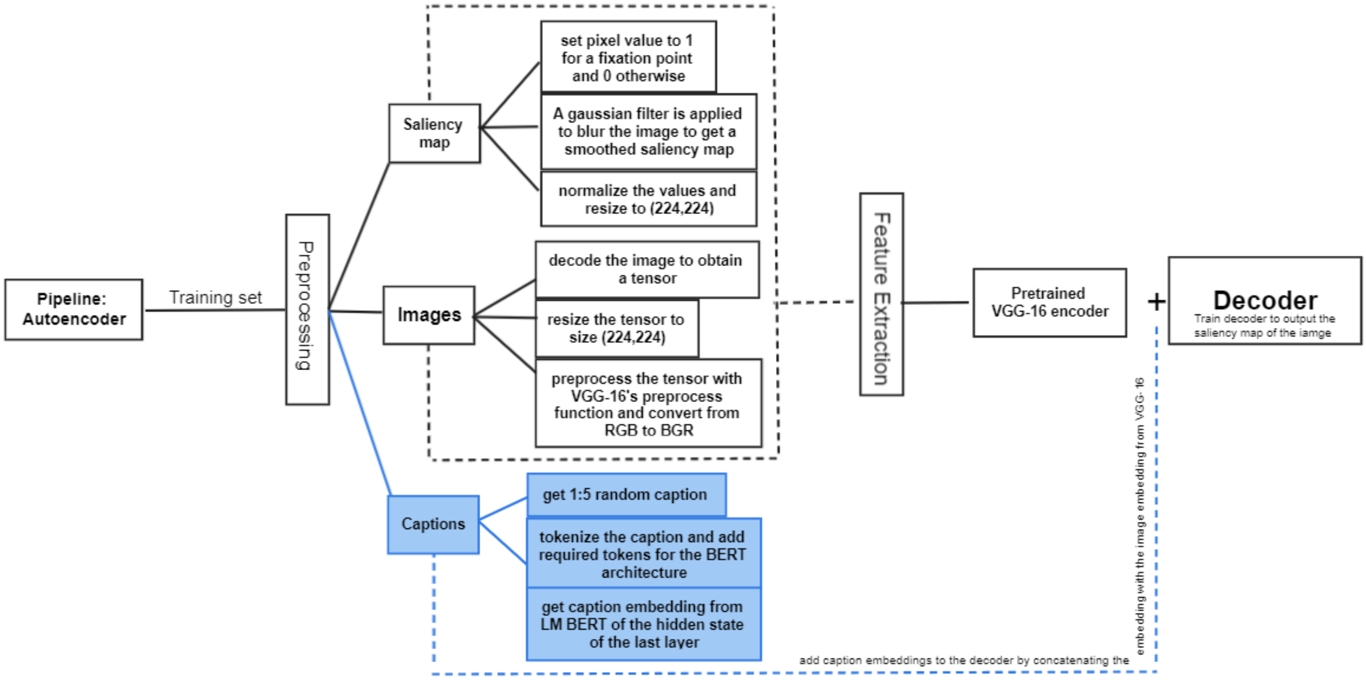

Pipeline: Autoencoder
Training set
Preprocessing
Saliency map
set pixel value to 1 for a fixation point and 0 otherwise
A gaussian filter is applied to blur the image to get a smoothed saliency map
normalize the values and resize to (224,224)
Images
decode the image to obtain a tensor
resize the tensor to size (224,224)
preprocess the tensor with VGG-16's preprocess function and convert from RGB to BGR
Captions
get 1:5 random caption
tokenize the caption and add required tokens for the BERT architecture
get caption embedding from LM BERT of the hidden state of the last layer
Feature Extraction
Pretrained VGG-16 encoder
+
Decoder
Train decoder to output the saliency map of the iamge
add caption embeddings to the decoder by concatenating the embedding with the image embedding from VGG-16

Fig. 3. Autoencoder Pipeline with added multimodal input

captions separately and bring them into a single dataset using a data generator. We chose to use a data generator for storage efficiency. For the same reason we also directly save this newly generated dataset to disk as a `TFRecords` dataset.

The images exist in different forms and need to be brought into a uniform shape for the encoder model. The VGG-16 model expects input images of the shape (224,224), wherefore we convert the images. The image is first reshaped to (224, 224, 3) and then preprocessed using the `vgg16.preprocess_input` function from `tf.keras.applications`, which converts the image from RGB to BGR and zero-centers each color channel color channel with respect to the mean values of ImageNet dataset, on which the original VGG-16 model was trained. So, if the images are preprocessed in this way, we can take full advantage of VGG16 as it has been optimized for this type of images.

The saliency maps must first be created from the fixation points of the 60 participants. Therefore, we first create an numpy array of zeros in the shape of the original input image to which the fixations belong. Each time a location appears for one of the participants, we add a 1 to the array at the same location. We then normalize the newly created saliency maps so that they are between 0 and 1. Before normalizing the values, a Gaussian filter is applied to the saliency maps. Finally, the saliency maps are converted to (224, 224, 1) since they are grayscale images and therefore only have one channel indicating the brightness of each pixel.

Preprocessing and tokenization of the captions is done by loading the the BERT and BERT preprocessing model and applying them to each caption.

Because BERT is bidirectional, the embedding of the `[CLS]` token, which is designated to be at the first position of each sentence, is encoded in such a way that it includes all the representative information of all tokens through the multi-layer encoding procedure. The representation of `[CLS]` is individual in different sentences, and can therefore be uses as a sentence encoding. Hence, we take the hidden state of the `[CLS]` token from the layer of the BERT model. This embedding is then fed into the dataset with a shape of (1,768).

The capgaze dataset [5] for validation is generated in a similar manner. Since eye-tracking was used for this study and values are recorded for the left and right eyes, we must first filter the values from the dominant eye. Apart from this, the other steps are largely the same as those for the COCO/SALICON dataset.

To be used in our models, we shuffle, batch and prefetch the data using the common TensorFlow methods.

A. The model

1) Baseline model: We use the SimpleNet architecture as a reference for our model, which is an encoder-decoder structure implemented in PyTorch. We provide a similar structure of their architecture that uses the VGG-16 model. VGG-16 is a CNN architecture that has been pre-trained on the ImageNet dataset and is commonly used for computer vision tasks. The encoder structure is loaded from `tf.keras.applications.vgg16` and we implemented

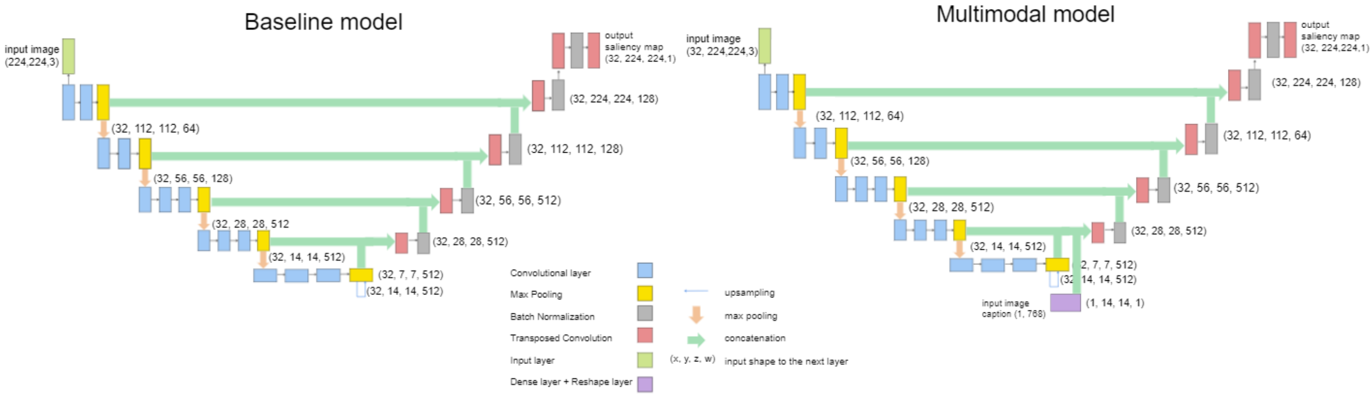


Fig. 4. A simplified representation the UNet structure of our model. The encoder block on the left side with Max Pooling layer for downsampling. The decoder block on the right (with concatenated input from the opposite encoder layer) to output the saliency map. The shapes indicated in brackets indicate the shape that is input to the next block.

the possibility to fine-tune a certain number of layers, setting the others to be non-trainable (transfer learning).

The Encoder consists of 19 layers, which we divided into 5 blocks. Extracted features are represented by its final layer. Each block has an input layer of shape (32, 224, 224, 3). The first two blocks additionally have two convolutional layers and a Max pooling layer, while the latter three have three convolutional layers followed by a Max pooling layer. After forwarding the input through all five encoder blocks, we use each block's output as input for our decoder.

The Decoder is consisting of 5 blocks. The first four blocks are implemented as follows: One transposed convolutional layer with a relu activation function and batch normalization at the end for stabilizing the model. The first block is using 512 filters, the second 256, third and fourth 128. Block 5 consists of two convolutional layers with relu activation, 128 filters for the first and sigmoid activation and one filter for the second layer, resulting in the same shape as the input. Batch normalization is applied between both layers.

We have adapted the UNet structure of SimpleNet into our TensorFlow model, which means that the encoder and decoder are symmetrically organized. Before every upsampling layer, there is a concatenation with the corresponding feature map from the encoder (shown as green arrows in Figure 4). We have adopted this UNet structure into our tensorflow model. In the final layer of the decoder, we integrate two convolutional layers. The saliency map is obtained from the final layer of the decoder.

B. Multimodal input

The model for the multimodal input is similar to the baseline model except for the integration of the additional image description. As the model must be able to process and understand natural language, we need to convert the text into numerical data. This is done through tokenization and vectorization, using the BERT model, as described above. For integrating the resulting sentence embedding in our model, we test different approaches. As a default, we add an additional dense layer to reduce the shape and a reshape layer to get a compatible shape for the sentence embedding and encoder output in order to concatenate it before feeding it into the decoder (see the right architecture in Figure 4). Additionally in this approach, the model would be able to adjust the dense layer's parameters during training. In the other approach, we simply bring the embedding in the right shape by tiling it and concatenate it with the encoder output as well. Both approaches have been used before as it can be seen in [1] and [10]. They should make it possible for our decoder to get information about the image and the image captions and we can investigate whether we find an improvement in performance.

C. Training, Testing & Evaluation

To evaluate and train the model we must provide the following parameters: the optimizer, the model itself, the dataset, the loss function and the number of epochs. We use Adam as the optimization algorithm a learning rate of 1e-4, since it is used by SimpleNet [13] or DeepGazeIII [7] and many other studies as well [4], [12]. The following reasons make the Adam optimizer a popular choice: By adaptively adjusting the learning rate for each parameter based on its past gradients, it converges faster. By adding momentum, Adam moves more efficient through the parameter space. When gradients are changing rapidly, it keeps the past direction to some degree, which helps to avoid getting stuck in a local minima. Compared to other optimization algorithms (e.g. SGD) it is computationally more efficient by using the moving average instead of keepting track of all past gradients. We still leave out the option to choose Adagrad or SGD. Our learning rate scheduler for Adam is set to 0, because we achieved good performances without scheduling the learning rate. The model is either the baseline model without image captions or the multimocal model, that includes these captions, which are trained on the SALICON dataset for training and testing, and on the capgaze dataset to evaluate the model on its ability to generalize. The inputs come in batches of size 32. During each epoch, the model makes predictions on the

TABLE I
RESULTS OF THE BASELINE MODEL AND MULTIMODAL MODEL REGARDING THE DATA THEY WERE TRAINED AND VALIDATED ON

	AUC	**CC**	**NSS**	**KLDiv**	**SIM**	**AUC**	**CC**	**NSS**	**KLDiv**	**SIM**
dataset	*basline model*					*multimodal model*				
trained on SALICON, validated on SALICON	0.9308628	0.84638095	0.72184545	0.14372084	0.7993543	0.932986	0.85283685	0.7171366	0.13897628	0.80406064
trained on SALICON, validated on capgaze1	0.86763364	0.679029	0.65899074	0.5427148	0.60021394	0.8665343	0.692144	0.6572099	0.50469744	0.6148439
trained on SALICON + capgaze1, validated on capgaze1	0.86446023	0.74304193	0.6897187	0.41196495	0.65951556	0.86780244	0.7646671	0.6810618	0.3993929	0.6653387
trained on capgaze1, validated on capgaze1	0.7409105	0.44526032	0.43316752	0.96303844	0.48806557	0.7755244	0.5217166	0.5184979	0.79649794	0.524302

images, calculates the loss and uses backpropagation to adjust the weights of the model using GradientTape for automatic differentation. Kullback-Leibler divergence VGG-16 computes the difference between our predicted saliency map and its ground truth map. The outcome defines how much information is retained from the predicted distribution and used in the model to update the weights. Zero is the desired outcome, with every larger value meaning that we are further away from the ground truth. Additionally, we included the opportunity of using regularization techniques in case of overfitting. For that, we implemented the option to apply L1-norm or L2-norm as kernel regularizers. L1-norm and L2-norm help to prevent overfitting by adding a penalty term to the loss function of a model. This penalty term discourages the model from fitting the training data too closely and instead encourages it to find a simpler solution that generalizes well to new data. We train our models for 10 epochs on the SALICON dataset (5 epochs took roughly one hour), then evaluated their performance on capgaze1 before finetuning them on this dataset for 5 epochs (started overfitting after that). Additionally we also trained both model architectures for 20 epochs directly on capgaze1. The results of that can be seen in Tab I

D. Compilation

To evaluate the models, 5 different metrics were used. The reason we selected these 5 metrics is their widespread usage and established reputation for evaluating saliency maps. They are all included in the MIT saliency benchmark, which is currently the gold standard for evaluating and comparing image-based saliency models.

From `tf.keras.metrics` we use the Area under ROC curve (AUC), which is explained below. In addition we have written custom metrics for the KLDiv divergence (see explanation above), Pearson Cross Correlation (CC), Normalized Scanpath Saliency (NSS) and Similarity (SIM) [2]. Together they cover distribution-based measures that compare smoothed prediction and fixation maps (CC, KLDiv, SIM) as well as location-based measures where some satistics are calculated at the fixation points (NSS, AUC). Except for KLdivergence loss, a high value is desired for all metrics.

- The most commonly used metric to evaluate saliency maps is the AUC metric, where pixels are treated as binary classifiers of a fixation that are positive when a certain threshold is reached. In this way, true and false positives can be measured.
- High values are desirable for NSS, which computes the averaged normalized saliency at fixated locations and penalizes false positives. It is invariant to linear transformations. This metric was specifically defiend for the evaluation of saliency models [11].
- The CC metrics computes the correlation between the normalized version of both saliency maps. Unlike the above, this statistical method penalizes positives and negatives equally. It ranges between 1 and 1, and a score close to 1 or 1 indicates a perfect linear relationship between the predicted saliency map and the fixation map.
- SIM measures the similarity between two distributions considered as histograms and the computed as sum of minimum values at each pixel. There is no overlap between two distributions when SIM gives a value of zero.

A good model should be good under a all of the mentioned evaluation metrics, as these measures reflect different aspects of the salience map.

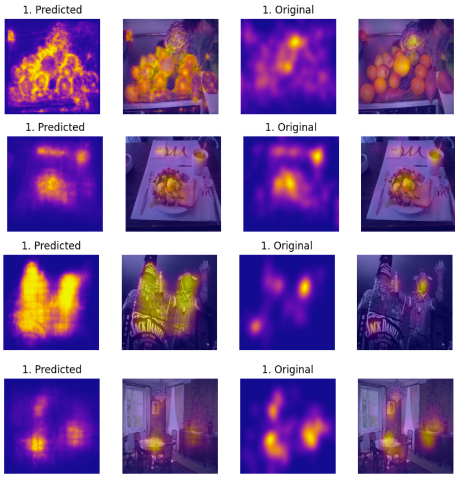


Fig. 5. Prediction results of our study: Upper two rows show the saliency map and saliency map with overlaid image in epoch 1 compared to epoch 9 for the multimodal model on the SALICON dataset. The last two rows show the saliency map and saliency map with overlaid image in epoch 1 compared to epoch 5 for the pretrained multimodal model on the capgaze dataset

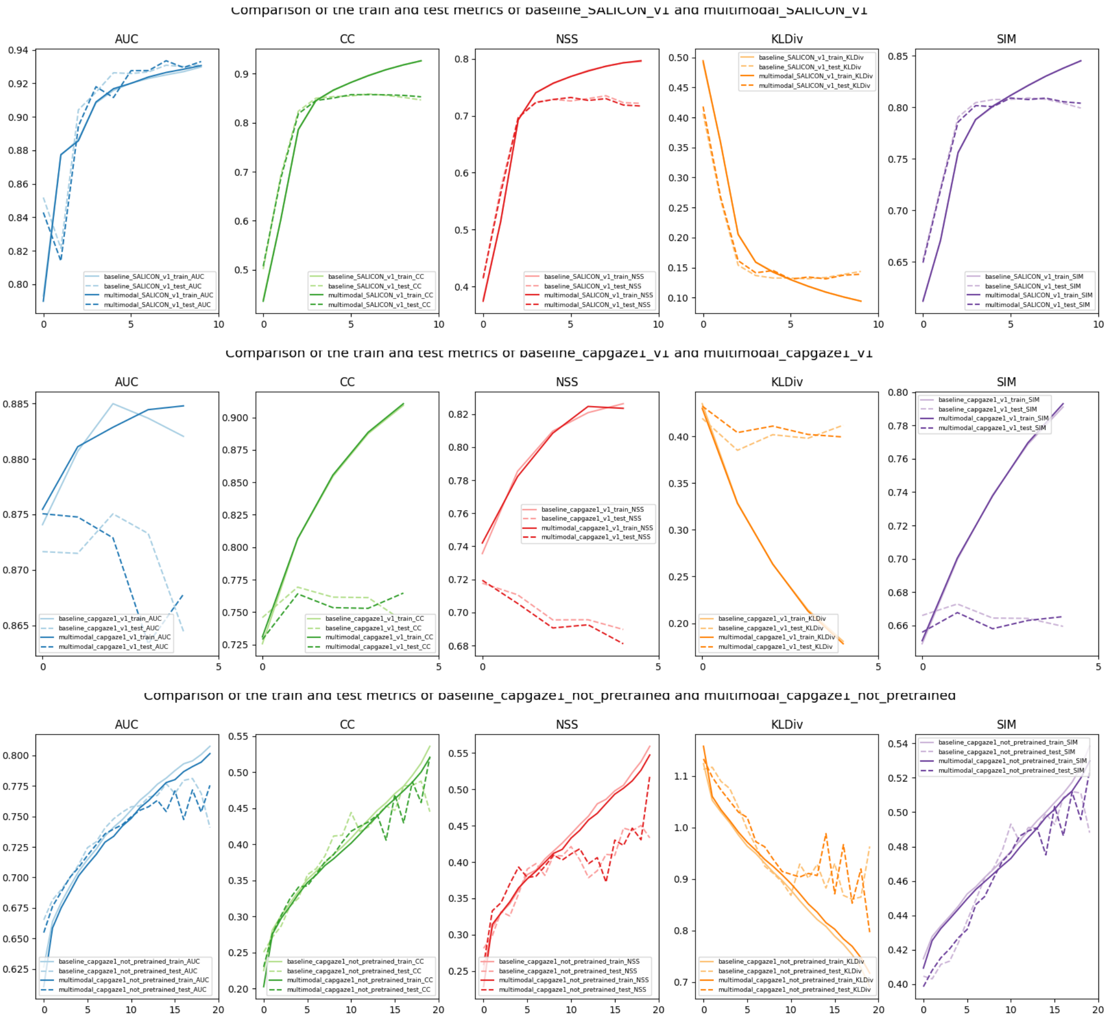


Fig. 6. Metric results of our study: Upper row represents the metrics of the SALICON dataset for the baseline and multimodal model. Middle row represents the metrics of the Capgaze dataset for the baseline and multimodal model with pretrained weights from the training on the SALICON dataset. Lower row represents the metrics of the Capgaze dataset for the baseline and multimodal model without pretrained weights.

IV. Results

We conducted experiments to compare the performance of a baseline saliency prediction model with a multimodal model that incorporated both image and text information. We compare the following set ups: Our main comparison was between the baseline and multimodal models trained and tested on the SALICON dataset. Contrary to our initial assumption, the metrics improved for both models nearly equally (see first row in Tab I and Fig 6). Both models reached values of 0.93 and for the other metrics the mostly differ around 0.01: The baseline model beeing slightly better in NSS and the multimodal scoaring the better results in the rest of the metrics. However, the metrics are not significant to suggest that adding text information improves the performance of the model.

We then evaluated the performance of the models on the Capgaze dataset. Both models generalized equally well on this dataset achieving almost identical results for every metric (see second row Tab I).

We then also trained both architectures with and without pretrained weights from the SALICON dataset. Without pretrained weights, the multimodal model achieved better performance on the test set after 20 epochs than the baseline model. With pretrained weights, the multimodal model generally performed better than without pertraining and also better than the baseline model except for the NSS metric. However, we would have needed more epochs to see whether we have improved significantly when adding image captions. Compared to their counterparts with loaded pretrained weights, we achieved much lower performance without pretraining (see last two rows row in Tab I and Fig 6).

Overall, our results show that adding text information does not necessarily improve the performance of the saliency prediction model. However, the pretrained weights from a related dataset can improve the performance of the model when generalizing on the Capgaze dataset. He *et al.* did improve the performance of their model when adding multimodal input. With our study we were not able to improve our models performance with additional image captions and are not found to guide the attention of the model.

V. Discussion

Our experiments provide some insights into the use of multimodal input for saliency prediction. Our findings suggest that incorporating text information does not necessarily lead to significant improvement in the model's performance. This may either be due to the nature of the datasets we used, which may not contain sufficient information in the image captions to guide the model's attention.

Another reason could be that the captions and saliency maps in the SALICON dataset are independent of each other, as the latter were created during a free-viewing task. This is supported by the fact that when both models were trained from scratch on capgaze1, where the saliency maps and captions were recorded simultaneously, the multimodal model performed slightly better than the baseline model, although they had previously performed almost identically in all the different test settings.

An additional factor might be that we introduced the caption embeddings too 'early' in the model and therefore lost the information. Other models that were also trained on multimodal inputs [1], [10] do not combine them until just before the last layer.

Our experiments also highlight the importance of pretrained weights for saliency prediction models. We found that using pretrained weights from a related dataset can improve the performance of the model. Several other limitations might have also affected our model's performance: We adjusted the shapes of SimpleNet to work in tensorflow and also trained on few epochs only due to time constraints and limited memory capacity.

Further research could explore the purpose of different feature maps of the image and image captions to better understand how they contribute to the saliency map. This could lead to the development of more effective multimodal saliency prediction models. In addition, our insights could be used to inform the design of more effective human-robot interaction systems, virtual reality experiences, and advertising campaigns. For example, saliency prediction models could be used to optimize the placement of visual elements in these contexts

A. *Where to find our model and the results:*

Our GitHub repository https://github.com/fynnomenon/iannwtf_project.git contains the datasets, model, and results, which can be loaded for reproduction.

References

[1] Mohammad Alsharid, Yifan Cai, Harshita Sharma, Lior Drukker, Aris T Papageorghiou, and J Alison Noble. Gaze-assisted automatic captioning of fetal ultrasound videos using three-way multi-modal deep neural networks. *Medical Image Analysis*, 82:102630, 2022.

[2] Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frédo Durand. What do different evaluation metrics tell us about saliency models? *IEEE transactions on pattern analysis and machine intelligence*, 41(3):740–757, 2018.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva: Exploring the limits of masked visual representation learning at scale. *arXiv preprint arXiv:2211.07636*, 2022.

[5] Sen He, Hamed R Tavakoli, Ali Borji, and Nicolas Pugeault. Human attention in image captioning: Dataset and analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8529–8538, 2019.

[6] Ming Jiang, Shengsheng Huang, Juanyong Duan, and Qi Zhao. Salicon: Saliency in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1072–1080, 2015.

[7] Matthias Kümmerer, Matthias Bethge, and Thomas SA Wallis. Deepgaze iii: Modeling free-viewing human scanpaths with deep learning. *Journal of Vision*, 22(5):7–7, 2022.

[8] Seung-Hyun Lee, Jang-Kyoo Shin, and Minho Lee. Non-uniform image compression using biologically motivated saliency map model. In *Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004.*, pages 525–530. IEEE, 2004.

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[10] Stuart J Miller, Justin Howard, Paul Adams, Mel Schwan, and Robert Slater. Multi-modal classification using images and text. *SMU Data Science Review*, 3(3):6, 2020.

[11] Robert J Peters, Asha Iyer, Laurent Itti, and Christof Koch. Components of bottom-up gaze allocation in natural images. *Vision research*, 45(18):2397–2416, 2005.

[12] Vasili Ramanishka, Abir Das, Jianming Zhang, and Kate Saenko. Top-down visual saliency guided by captions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7206–7215, 2017.

[13] Navyasri Reddy, Samyak Jain, Pradeep Yarlagadda, and Vineet Gandhi. Tidying deep saliency prediction architectures. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10241–10247. IEEE, 2020.

[14] Boris Schauerte and Rainer Stiefelhagen. "look at this!" learning to guide visual saliency in human-robot interaction. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 995–1002. IEEE, 2014.

[15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[16] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Collecting image description datasets using crowdsourcing. *arXiv preprint arXiv:1411.3041*, 2014.

[17] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.