Privileged Attribution Constrained Deep Networks for Facial Expression Recognition Implementation

AL OZAIBI Youssef YOUSSEF.AL_OZAIBI@ETU.SORBONNE-UNIVERSITE.FR
Master Ingénierie des Systèmes Intelligents
Sorbonne Université
Paris, France

DJAMAI Hania HANIA.DJAMAI@ETU.SORBONNE-UNIVERSITE.FR
Master Ingénierie des Systèmes Intelligents
Sorbonne Université
Paris, France

SELLAMI Amine AMINE.SELLAMI.1@ETU.SORBONNE-UNIVERSITE.FR
Master Ingénierie des Systèmes Intelligents
Sorbonne Université
Paris, France

Editor: https://github.com/yozaii/Privileged-Attribute-Loss-Implementation

Abstract

Facial Expression Recognition is a widely researched subject and can be applied to a wide variety of domains. Using Priveleged Attribution Loss (PAL) in deep models, state-of-the-art results have been achieved when predicting the seven basic emotions (sadness, angriness, happiness, surprise,disgust, fear, neutral) from an image. The article's method redirects the model into focusing on pertinent regions of the face during training phase to have a better performance during prediction. In our work, we will go through the details on how to implement the algorithm to an existing deep backbone model to have a functioning program. The main challenge can be thus described as how to interpret the algorithms of the referenced PAL article into functional code. This involved building the model architecture, incorporating the PAL loss into our training loop, loading large datasets for training, and evaluating the model. The primary difficulty is getting our model to match the state-of-the-art performance on the given datasets, RAF-DB and AffectNet.

Keywords: FER, PAL, RAF-DB, AffectNet

1. Introduction

Our main goal in this project was to implement a practical application of Bonnard et al. (Jun, 2022) Privileged Attribution Constrained Deep Networks article. In other words, creating a model following the original article's algorithms that can be trainable given a dataset, and can predict a given image's emotion with a decent accuracy. To achieve this goal, we had to have a good understanding of the article's algorithms and ideas. In addition, using Machine Learning frameworks, we had to learn how to interpret the algorithms to create a model in functioning code, load and process large datasets, and evaluate the performance of our code.

To demonstrate our understanding of the reference article, we will delve into its details in the following section(2). The reference article does a good job in explaining the details of the algorithms, nevertheless we will still try to provide deeper explanation and more intuition regarding the algorithms when it is possible to elaborate further.

In section (4), we will show how we incorporate the article's main idea, the Privileged Attribution Loss (PAL) in our training loop. We will also show how we handle and process large amounts of data, and how we evaluate the performance of the model.

2. Presentation of the algorithm

The main idea behind the article's method is adding a Privileged Attribution Loss (PAL) using facial landmark heatmaps to the model's loss during the training phase, thus guiding the model to focus on the pertinent sections of the face such as the eyes, mouth, and nose. Existing deep learning models are used as backbones (ResNet or VGG16), and their loss functions are modified to account for the new method. Three main points constitute the creation of PAL : facial landmark heatmaps, attribution gradient method, and the PAL algorithm itself.

2.1 Heatmaps

First, a facial alignment model is used on a 2D image to create landmarks, which are then passed through a filtering algorithm to generate each image's corresponding heatmap. In our work, we do not use a model to create the landmarks (Dapogny and Bailly (Apr, 2020)), but instead they are already provided through an existing database. Our work comprised of generating the heatmaps from the landmarks files. Each landmarks file in the database contains 68 rows of two numbers, corresponding to the x and y coordinates of a landmark in an image. Thus we have our landmark vector L with a shape of [68 * 2]. The heatmap image h is first initialized as with all its pixels set to zero (black). Afterwards, for each landmark coordinate in our landmark vector L, we set the corresponding heatmap pixel value to 1 (white).

$$h(i,j) = \begin{cases} 1, & \text{if } i,j \in \text{L} \\ 0, & \text{otherwise} \end{cases}$$

Finally, we convolve a gaussian filter with a standard deviation of σ with our heatmap image to get the final heatmap output :

$$\begin{aligned} &\text{Gaussian kernel} : g(i,j) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{-(i^2+j^2)}{2\sigma^2}) \\ &h = h * g \end{aligned} \tag{1}$$

Contrary to the original article, we have chosen to use the filtering operation over the entire heatmap in one go instead of passing the gaussian filter over each landmark seperately and then getting the sum of each landmark filter to produce the final output. This decision was made in the interest of execution time, since we have noticed that visually, the difference between the original filtering method and ours does not show any substantial difference.

2.2 Attribution gradient method

There are two attribution gradient methods proposed by the article. *Grad* and *Grad*Input*. For the *Grad* method, we calculate the derivative of the sum S of the absolute value of the output emotion vector f_o(classifier) with respect to the outputs of an intermediate layer. The outputs of the intermediate layers are the pixels of the convolutional layers after a forward pass of an input to the model. So to calculate the attribution a^L of a layer L, we calculate how a small change in each pixel of L with coordinates i, j and channel c will influence the output vector ($f^L_{i,j,c}$) :

$$S = \sum_{i=1}^{7} f_o(i)$$
$$a^L_{i,j,c} = |\frac{\partial S}{\partial f^l_{i,j,c}}| \tag{2}$$

*Grad*Input* works similarly except we take into account the input to the attribution layer :

$$a^L_{i,j,c} = |\frac{\partial S}{\partial f^l_{i,j,c}}| \cdot f^l_{i,j,c} \tag{3}$$

As highlighted by the article, the "intuition of the difference between the two o is that *Grad* corresponds to how a small change in the input will impact the output of the network, whereas Grad*Input corresponds to the total contribution of a feature on the output of the network." Even without using PAL, the output of the high level layers of a deep learning model will roughly start to focus around areas slightly similar to the facial heatmaps we provide. Therefore by using the heatmaps and the *Grad*Input* method during the calculation of the PAL (see 2.3), we exaggerate the focus around the landmarks even more since we take high-level feature and cross-correlate it with the heatmaps.

2.3 Privileged Attribution Loss

As discussed in the last section, we try to find the cross-correlation between the attribution and the heatmaps and maximise this value. $\mu(a^L) = \sum_{i,j} a^L_{i,j,c}$ and $\sigma^2 = \sum_{i,j}(a^L_{i,j,c} - \mu(a^L))(a^L_{i,j,c} - \mu(a^L))$. The PAL is thus calculated as :

$$PAL = -\sum_{i,j,c} \frac{a^L_{i,j,c} - \mu(a^L)}{\sigma(a^L)} * a^*_{i,j,c} \tag{4}$$

Where $a^*_{i,j,c}$ is the heatmap. This loss gets added to the total loss (categorical crossentropy in our case) during the training phase only to guide the attribution layer. The attribution layer is also guided depending on the channel strategy. There are three main strategies : all channels, mean, and mean of half. All channels means that the PAL will be calculated with respect to the attribution across all its channels C.

$$a^L_{i,j} = \sum_{c=1}^{C} = a^L_{i,j,c} \tag{5}$$

This does not reduce the values of the attribution in any way, and applying the PAL using this strategy can constrain the model from making other spatial connections and instead force it to overfocus on the heatmaps. Therefore we add a weighting parameter $\frac{1}{C}$ to reduce the values of the attribution and thus reduce the value of PAL with respect to the total loss. This channel strategy is the mean strategy.

$$a^L_{i,j} = \frac{1}{C}\sum_{c=1}^{C} = a^L_{i,j,c} \tag{6}$$

The mean strategy still constrains all channels of the attribution layer to focus on the heatmaps, albeit to a lesser degree than the all channels strategy. To find a balance between focusing on the heatmaps and learning other spatial information during training, we will only use half of the channels of the attribution channel ($C_1 = \frac{C}{2}$) to calculate the attribution layer and weight them with C_1.

$$a^L_{i,j} = \frac{1}{C1} \sum_{c=1}^{C1} = a^L_{i,j,c} \tag{7}$$

3. Data

To train our models, two Facial Expression Recoginition (FER) datasets are used : *RAF-DB* and *AffectNet*.

RAF-DB a train set of 12271 aligned images, and a test set of 3068 aligned images. An aligned image means that the location of the eyes, nose, and mouth are roughly on the same level vertically and horizontally across the dataset. This will allow us to use a simple model architecture without having to account for translation variance. All images are annotated with their respective emotion out of the 7 possible emotions.

AffectNet also provides a similarly annotated aligned image dataset. For the 7 possible emotions, 280k training images and 3.5k test images are given.

4. Evaluation

4.1 Description of the experiment

The architecture of our model is relatively simple, the main part of the experimental phase involved modifying the training loop of the model to implement the PAL. First, we used either VGG16 or ResNet50 as our backbone in the model architecture, and then we added a classifier top comprised of two fully connected layers and a softmax activated output vector with 7 classes corresponding to the seven basic emotions.

In our custom training loop, we take batches of data containing the input images and their associated heatmaps. Afterwards, we calculate the categorical crossentropy loss from the difference between our predicted output, *y_pred*, and the ground truth label y. Two gradients are then calculated, the *Grad*Input*, and the back propagation gradient using keras's gradient tape. The *Grad*Input* allows us to obtain the attribution of a chosen layer and thus calculate the PAL loss. Finally we add this loss to the categorical crossentropy loss to have a final total loss, which is then used to calculate the back propagation gradient by using an optimizer. We have chosen an Adam optimizer with a 1e-4 learning rate during our experiments.

To handle large amounts of data coming from the datasets, we only load the filenames to the short term memory. We then use a data generator class by importing from keras.utils.Sequence. By using that class, we only load batches of images and heatmaps to the short term memory at every train step, thus allowing us to train our model on very big datasets.

To incorporate multiple channel strategies into our model, we have created a variable during our model's class initialization that indicates the channel strategy we use. 0 indicates using all channel attribution, 1 is mean channel attribution, and 2 is half of mean channel attribution.

4.2 Results

Unfortunately, we have not been able to get concrete results to compare with the article. We were only able to verify that our model was converging towards a value since our categorical precision kept increasing, but are unable to provide a final answer in time.

4.3 Discussion

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

5. Conclusion

This project has given us an opportunity to learn the Tensorflow and Keras frameworks at a deeper level. It also allowed us to try and apply scientific articles into executable code. The main difficulties we have faced were managing large datasets, understanding gradients in the training loop to apply the privileged attribution loss, and understanding how to manage matrix operations when information comes in batches. Although our final result was far from ideal and our model was not able to integrate the privileged attribution loss, it was a very interesting project and putting things together was definitely satisfying.

6. Bibliographie

Références

J. Bonnard, A. Dapogny, F. Dhombres, and K. Bailly. Privileged attribution constrained deep networks for facial expression recognition. Jun, 2022.

E. Arnaud A. Dapogny and K. Bailly. Tree-gated deep mixture-of-experts for pose-robust face alignment. *IEEE Transactions on Biometrics, Behavior, and Identity Science, vol. 2, no. 2, pp. 122–132*, Apr, 2020.