APPLIED MACHINE LEARNING SYSTEM ELEC0134 22/23 ADDITIONAL MATERIAL

SN: 22157928

ABSTRACT

This document contains additional material related to the Applied machine learning system ELEC0134 22/23 report.

1. FACE DETECTOR AND LANDMARKS PREDICTOR

This section describes in detail the face detection and landmark prediction stage used in all the approaches. The face detection stage is performed using a combination of the feature descriptor HOG (Histogram of Oriented Gradients) and the classification algorithm SVM [1]. The HOG algorithm counts occurrences of gradient orientation in the localized portion of an image. In particular, the HOG algorithm is composed of several steps:

- Resize the input image to $64x128$ pixels and divide it by $8x8$ patches
- For every patch and for every pixel of the patch calculate the magnitude and the angle of the gradient in the following way:

$$Magnitude(\mu) = \sqrt{G_x^2 + G_y^2} \tag{1}$$

$$Angle(\theta) = |tan^{-1}(G_y^2/G_x^2)| \tag{2}$$

where

$$G_x(r,c) = I(r,c+1) - I(r,c-1) \tag{3}$$

$$G_y(r,c) = I(r-1,c) - I(r+1,c) \tag{4}$$

and r, c and $I(r,c)$ represent respectively the row, the column and the pixel intensity.

- A 9 bin histogram is constructed, where every bin represents a range of 20 degrees (e.g. 0-20 or 40-60) covering all the degrees from 0 to 180. In particular, the value of every bin is calculated in the following way: For every pixel of the patch the following values are calculated

$$j = \left(\frac{\theta}{\Delta\theta} - \frac{1}{2}\right) \tag{5}$$

$$V_j = \mu\left(\frac{\theta}{\Delta\theta} - \frac{1}{2}\right) \tag{6}$$

$$V_{j+1} = \mu\left(\frac{\theta - C_j}{\Delta\theta}\right) \tag{7}$$

where C_j is the center value of the range of j^{th} bin, $theta$ is angle of the gradient calculated in the previous steps and $\Delta\theta$ is the step size used to construct the bins (in this case $\Delta\theta = 20$). Every V_j is summed up to the value of the j^{th} bin.

- Once the previous computation is completed, we have a 16x8x9 feature matrix (9x1 feature vector for every 8x8 patch). At this point, from the feature matrix, 4 9x1 feature vectors are clubbed together to form a new 2x2 block. This clubbing is done in an overlapping manner with a stride of 8 pixels and for all 4 cells in a block, we concatenate all the 9 point feature vectors for each constituent cell to form a 36 feature vector.
- Values of every 36 feature vectors are normalized using the L2 norm:

$$k = \sqrt{b_1^2 + b_2^2 + ... + b_{36}^2} \tag{8}$$

$$f_i = \left[\left(\frac{b_1}{k}\right), \left(\frac{b_2}{k}\right), ..., \left(\frac{b_{36}}{k}\right)\right] \tag{9}$$

where f_i is the normalized feature vector and $b_1, ..., b_{36}$ are the values of the unnormalized feature vector. This normalization step is required to make the algorithm more robust again local variations of brightness and contrast.

- Once the previous step is completed we have a 7x15x36 feature vector that represents the HOG features.

The HOG algorithm is applied on the input image in a sliding window manner and the features extracted in every window are then passed to an SVM that classifies it as face or non-face. The result is a heatmap of the input image where the value represents how likely it is that that pixel is part of a face and using a threshold we extract the pixels of the face. Furthermore, to capture the object of interest (in this case a face) at different scales the algorithm uses a image pyramid method: it resizes the input image at different sizes to be able to capture different scales of the object with the same sliding window. Finally, the face landmarks prediction is performed using ERT (Ensemble of Regression Trees), as in [2], that uses the gradient tree boosting algorithm in combination with the sum of square error loss function.

2. REGULARIZATION

The following section contains a detailed description of the regularization techniques used in the CNN approach. The regularization techniques are the following:

- Early stopping: In [3] is described as one of the most used regularization techniques, thanks to its effectiveness and simplicity. The authors explain how we can also consider this technique as a hyperparameter selection algorithm where the hyperparameter is the number of epochs. Furthermore, the authors explain how $\epsilon\tau$, where ϵ is the learning rate and τ is the number of epochs, is a measure of the effective capacity of the model as it limits the volume of the reachable parameter space, and in this sense behaves as the reciprocal of the coefficient used in L2 regularization.

- L2 regularization: with this regularization technique, the loss function is extended by adding a regularization term Ω defined as

$$\Omega(\theta) = \sum_{i=1}^{n} {\theta_i}^2 \tag{10}$$

  where θ are the parameters of the model. In particular, the regularization term prevents the weights of the model from growing, penalizing high values.

- Dropout: this technique consists in randomly "shutting down" neurons during training. In particular, when this technique is used during training, random neurons are set to zero and in this way, the model is forced to not rely on few features but maintain a balance between the features. This technique was only used on the last fully connected layer of the model.

- Stochastic Depth: similar to dropout, stocastic depth sets to zeros entire layers. The overall effect of this technique, apart from regularizing, is that it makes easier to train deeper models. In particular, using stocastic depth during training is virtually the same as having a less deeper and easier model to train, while maintaining the same number of layers, hence the same complexity and expressive power.

3. TRANSFER LEARNING

Transfer learning (TL) relates to exploiting the knowledge acquired in one task to improve the performance in another one. In a practical sense, it refers to reusing a model developed and trained on a task as a starting point for a new task. TL solves the problem of how to initialize the weights of a model. In particular, several kinds of initializing techniques exist for the weights of a model, the most simple one is assigning them in a random fashion. The problem is that the performance of the model could be highly penalized by a bad initialization. TL gives the possibility to exploit the weights learned in a previous task, which can represent a good starting point for the new task, especially in DL models, where the first layers of the NN (or as in this case a CNN) represent low-level features, that contrary to the high-level features, are not task-specific and turn out to be useful across different tasks. In this work, as mentioned before, the weights exploited are the ones pretrained on ImageNet-1K a subset of 1.2M images of the original ImageNet dataset [4] labeled with 1000 classes. The advantage that these weights give as is represented by the features learned on a huge benchmark dataset, a training that would be either impossible or computationally and from a time point of view very costly. In particular, as mentioned before, the features that result more useful in a transfer learning framework are the low-level features that are not task-specific, that in this case are the low-level features of an image like points, edges, shapes and even higher level features.

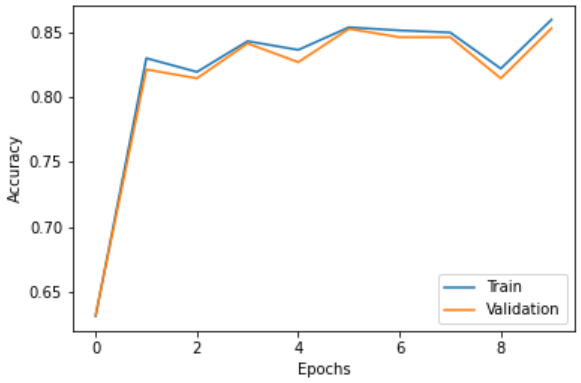
The Fig. 1 shows evidence of the advantage that pre-training gives. In particular, the two plots show the training and testing accuracy of the same model but with and without pre-training and the model without pre-training reaches 0.8 accuracy after 10 epochs, whereas the pre-trained is over 0.8 after the first epoch.

4. ADDITIONAL RESULTS

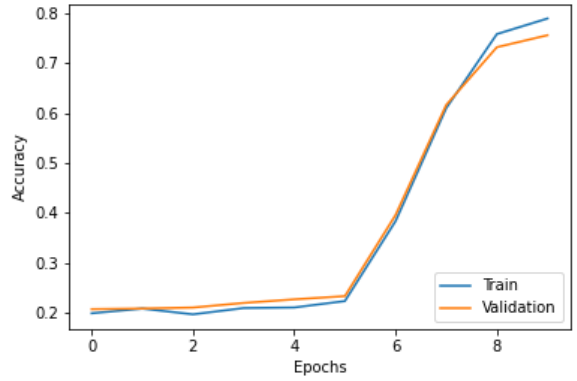
The Fig. 2 shows the results of performing k-fold cross validation on the CelebA dataset and the KNN algorithm. In particular, it shows how the best k is in the range between 20 and 40. The Fig. 3 shows evidence of the fact that the presence of images with dark glasses makes their classification impractical. In particular, the last column shows that the label 4 (brown/dark eyes) is the one that is most predicted and also that the wrong predictions are more or less equally distributed in the other labels, and this is caused by the fact that the images with dark glasses are mistaken for the images with dark eyes. A similar thing happens in the face shape recognition task, where the presence of a beard on the cartoon makes the classification much more complex if not impractical. As in Fig. 3, the thing happens in Fig. 4. Similarly, the cause is probably to attribute to the fact that many images of label 4 present a beard. Furthermore, the the Fig. **??** show respectively the confusion matrix of the most performing approaches on the gender detection and on the smile detection tasks.

5. ADDITIONAL ILLUSTRATIONS

This section contains additional illustrations that complement the report. The Fig 7 shows the difference between a depthwise separable convolution and a conventional convolution. On the other hand, the Fig. 8 shows the difference between



(a) Plot of the training accuracy and of the validation accuracy of the EfficientNet with pre-training.



(b) Plot of the training accuracy and of the validation accuracy of the EfficientNet without pre-training.

Fig. 1: Comparison between the pre-trained model and the model without pre-training of the face shape recognition task.

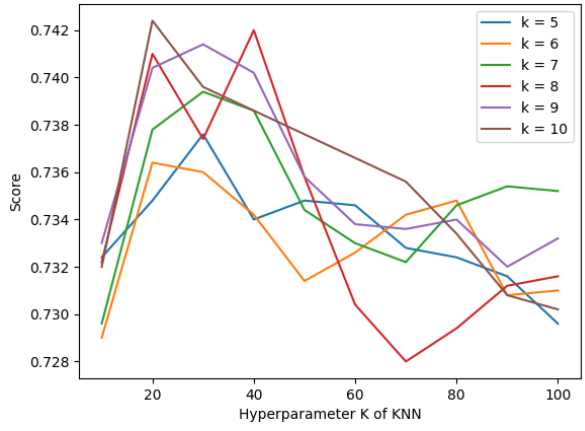


Fig. 2: K-fold cross validation results, on the the CelebA dataset and the KNN algorithm. The k in the legend refers to the parameter of k-fold cross validation.

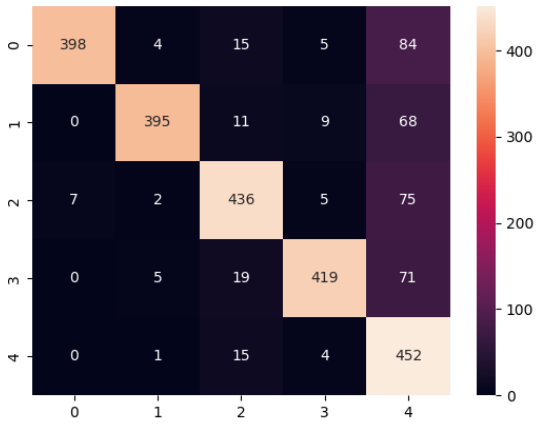


Fig. 3: Confusion matrix of static cropping + SVM approach on the eye color recognition task.

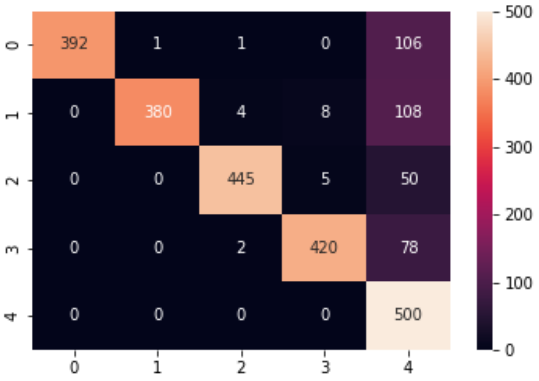


Fig. 4: Confusion matrix of CNN approach on the face shape recognition task.

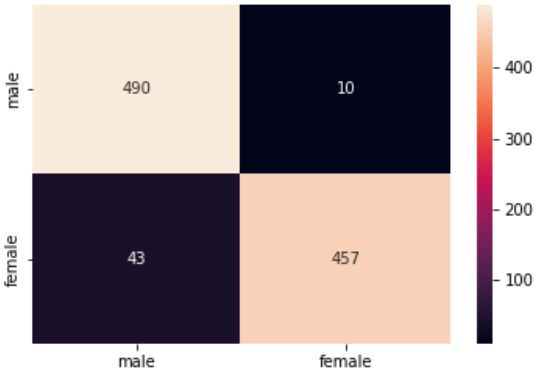


Fig. 5: Confusion matrix of CNN approach on the gender detection task.

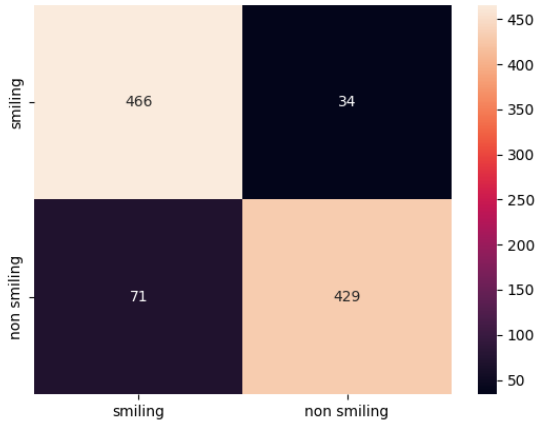


Fig. 6: Confusion matrix of the Landmarks + KNN approach on the smile detection task.

a Bottleneck and an Inverted Bottleneck. Both the depthwise convolution and the Inverted Bottleneck, are modules part of the CNN used.

6. REFERENCES

[1] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Ieee, 2005, vol. 1, pp. 886–893.

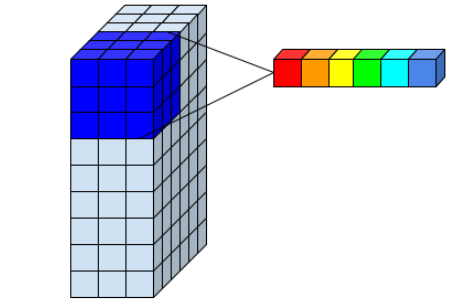
[2] Vahid Kazemi and Josephine Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1867–1874.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT press, 2016.

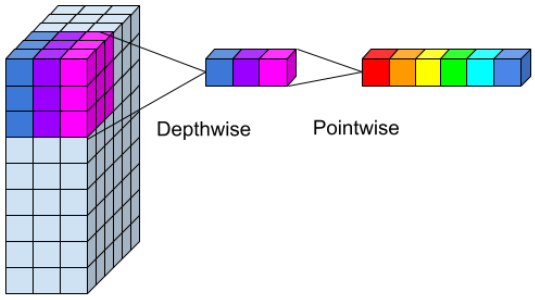
[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[5] Marc Päpper, "Depthwise separable convolutions in pytorch," `https://www.paepper.com/blog/posts/depthwise-separable-convolutions-in-pytorch/` February 06, 2021 (accessed January 11, 2021).

[6] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

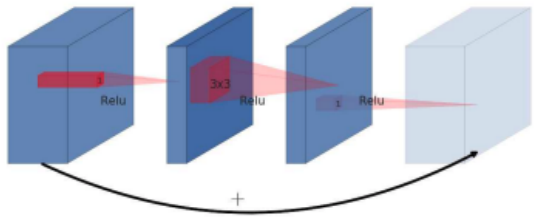


(a) Conventional convolution

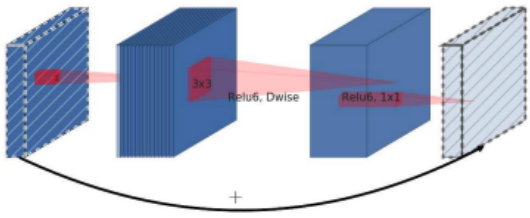


(b) Depthwise separable convolution

Fig. 7: Illustration of the difference between normal convolution and a depthwise separable convolution [5].



(a) Bottleneck



(b) Inverted Bottleneck

Fig. 8: Illustration of the difference between a Bottleneck and an Inverted Bottleneck [6].