Project 3: ECE 763

1st Arpad Voros
Electrical and Computer Engineering
North Carolina State University
Raleigh NC, USA
aavoros@ncsu.edu

2nd Bryce Abbott
Mechanical and Aerospace Engineering
North Carolina State University
Raleigh NC, USA
tbabbott@ncsu.edu

3rd Jonathan Service
Electrical and Computer Engineering
North Carolina State University
Raleigh NC, USA
jmservic@ncsu.edu

***Abstract*—This paper presents a deep learning model for face detection. The babysitting method was implemented to optimize the regulation and learning rate for the model with a coarse and fine search. Face data was cherry-picked, imported, and preprocessed to improve performance. The simple LeNet-5 architecture was selected as the initial network structure, but a hand-crafted network was also implemented that provided improved accuracy. It was found that the babysitting process enabled a higher accuracy to be achieved compared to simply choosing common hyper-parameter values. This paper also discusses the effect of dropout on overfitting and accuracy.**

***Index Terms*—CNN, computer vision, facial recognition**

I. Methodology

To create the deep learning model for face detection, the Keras API (built on Python's TensorFlow 2.0) was used in the Google Colab environment. Cropped face and non-face images used in Project 1 & 2 were reused for this project using the FDDB dataset. These images are 3-channel RGB at a resolution of 20×20 pixels. Image sets were stored in a single array with size $\mathrm{B} \times \mathrm{H} \times \mathrm{W} \times \mathrm{C}$, where B is the number of batches, H is the image height, W is the image width, and C is the number of channels (or colors). Images were downscaled by 255 to get a floating point value for each datapoint (pixel), and preprocessed to make the set more comparable. The data labels were one-hot encoded, using a set of binary variables to support more than two categories if desired.

A. Architecture

A simple LeNet-5 convolutional neural network architecture was selected for its simplicity and wide use in image recognition. The structure of this model is shown in Table I.

TABLE I: Model Training Metrics

Layer (type)	Output Shape	Param #
Conv2D	(None, 20, 20, 6)	456
Dropout	(None, 20, 20, 6)	0
AvgPooling	(None, 10, 10, 6)	0
Conv2D	(None, 6, 6, 16)	2416
Dropout	(None, 6, 6, 16)	0
AvgPooling	(None, 3, 3, 6)	0
Flatten	(None, 144)	0
Dense	(None, 120)	17400
Dropout	(None, 120)	0
Dense	(None, 84)	10164
Dropout	(None, 84)	0
Dense	(None, 2)	170

This architecture will be the target discussion of the paper.

II. Data Pre-processing

Differences in image parameters like lighting and contrast could negatively affect the training and test accuracy, so the images were then zero-centered and normalized to provide a more comparable data set [1]. To zero-center the data, the mean with respect to the number of images was subtracted from the pixel value of each image for each channel. For normalization, the resulting array was divided by the standard deviation with respect to the number of images. These operations reduce the amount of variation due to contrast and intensity changes, which enables reliable comparison between features for repeatable detection. In addition, the images were allowed to be rotated up to 20 degrees, and horizontal and vertical flips were randomly applied to some of the training images for robustness.

III. Babysitting the Learning Process

A. Hyperparameter Optimization

The babysitting process enabled optimization through the control of network hyperparameters, which influence loss and the resulting accuracy. After building the model, initial efforts were spent to determine reasonable learning rates and regularization values, while performing sanity checks to validate the model. First, the model was run to check if the initial loss was reasonable. Then, with a small regularization (L1 and L2) value (10^{-6}), a learning rate was found that made the loss decrease without exploding. It was concluded that a learning rate between 10^{-3} and 10^{-6}, and a L1 & L2 regularization between 10^{-5} and 10^{5} should provide the best results. To determine the optimal hyperparameters, a coarse and fine search were used sequentially.

1) Coarse Search: For the coarse search, temporary values for regularization and learning rate were selected uniformly at random and tested for a total of 100 trials. The helped to narrow down the search to a learning rate between 10^{-4} and 10^{-3}, and a regularization between 10^{-5} and 10^{-3}. A summary of the coarse search results is shown in Fig. 5 with influential learning rates and regularization values are enclosed by red rectangles.

2) Fine Search: For the fine search, the narrower bounds were used with the same uniform random sampling. With a total of 100 trials, it was found that a learning rate of 9.4×10^{-4} and a regularization of 7×10^{-4} provided the highest accuracy. These values will be used for the following implementation and results. A summary of the fine search results is shown in Fig. 6 with influential learning rates and regularization values are enclosed by red rectangles.

IV. Model Implementation and Results

The LeNet-5 network was trained with 3000 of both face and non-face training images. The results at each step of the convolution network are shown for an example face image in Table V [2]. It should be noted, that these visuals show all convolutional, pooling, and drop out layers. After these steps, the model is fed through multiple dense layers (which are not visualized). One interesting observation that was made was the visual of the last layer looks similar between the three positive images, whereas the two negative images have significantly different distributions.

Figure 1 shows the training progress over time. The model trained on pre-processed images (referenced in II) provided a 93.59% final training accuracy and a 96.88% validation accuracy after 25 epochs.

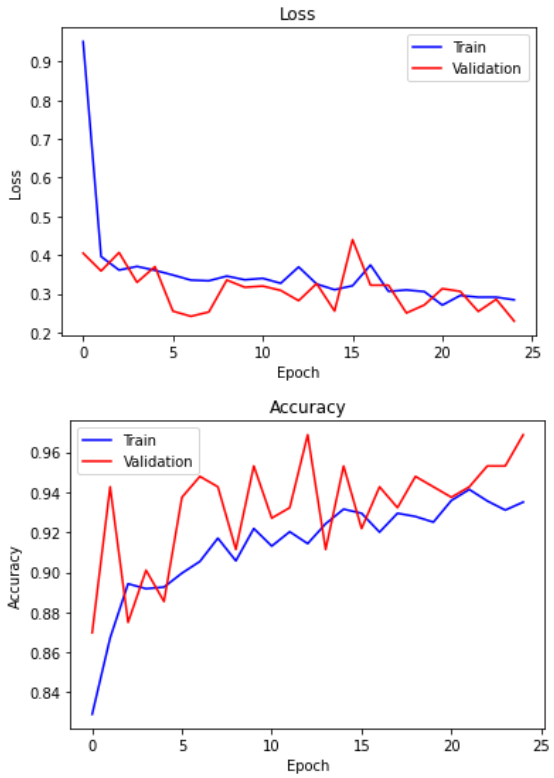


Fig. 1: LeNet-5 architecture trained on pre-processed data

A. Effect of pre-processing

Figure 2 shows the training progress over time without pre-processing. The model only provided a 92.88% final training accuracy and a 93.23% validation accuracy, which is 3.68% lower than that with pre-processing. This shows how helpful pre-processing is for success.

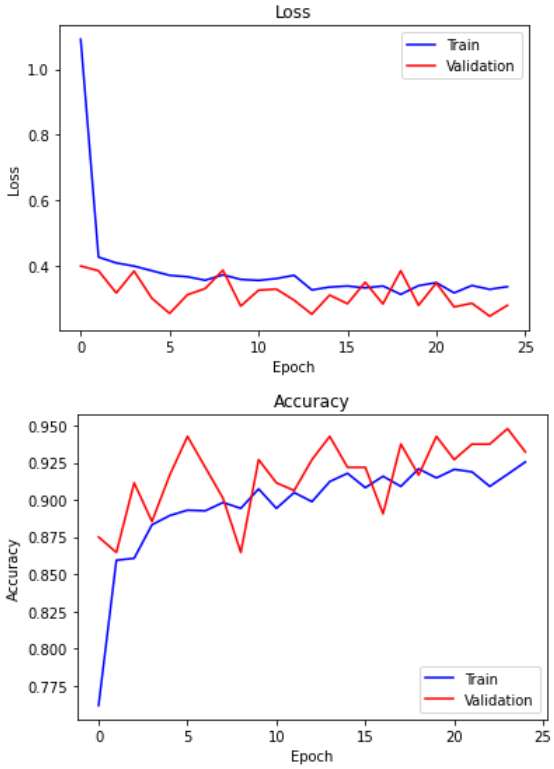


Fig. 2: Training History Without pre-processing

It should be noted that due to the naturally small size of this LeNet-5 (∼32,000 parameters), the training and validation accuracy is more sporadic as small changes during parameter updating in back-propagation affects the entire model at a greater ratio than a model with more parameters. Nevertheless, these results for both models trained on nonpre-processed and pre-processed data, respectively, are incredibly good and efficient considering how small the model is.

B. Effect of dropout

Dropout is a regularization technique where randomly selected neurons are ignored during training. This means that other neurons will have to handle the representation required to make predictions for the missing neurons. This gives neurons less opportunity to develop weights that are highly specific to noise in the training data, and the network becomes less sensitive to the specific weights of the neurons. This reduces the likelihood of over-fitting, and creates a more generalized model with a goal of achieving a higher validation accuracy.

To determine the best dropout rate (with a rate of 1 being all neurons ignored and 0 being all neurons used), the validation accuracy for a sweep of rates were studied. The accuracy corresponding to dropout rates tried in increments of 0.1 are shown in Fig. 3. The dropout had a positive effect on the validation accuracy at low rates. At higher dropout

rates, however, validation accuracy decreased due to fewer neurons being available. A dropout rate of 0.4 was the most effective and provided an additional 1% validation accuracy. This improvement in fitting to the test data supports the idea that dropout can reduce over-fitting to training data, especially for models with small amounts of training data.

```
dropout: 0.0, val_accuracy: 0.9340000152587891
dropout: 0.1, val_accuracy: 0.9419999718666077
dropout: 0.2, val_accuracy: 0.9319999814033508
dropout: 0.3, val_accuracy: 0.9340000152587891
dropout: 0.4, val_accuracy: 0.9440000057220459
dropout: 0.5, val_accuracy: 0.9200000166893005
dropout: 0.6, val_accuracy: 0.9139999747276306
dropout: 0.7, val_accuracy: 0.9179999828338623
dropout: 0.8, val_accuracy: 0.8859999775886536
dropout: 0.9, val_accuracy: 0.8700000047683716
```

Fig. 3: Effect of Dropout Rate on Validation Accuracy

V. Evaluation

There were two LeNet-5 models trained, as well as another model introduced in section VI-A. The following table shows the binary accuracy of each model on pre-processed data, given by different datasets produced by each author.

TABLE II: Binary Accuracy

Test Dataset	Model Name **LeNet-5 1**	**LeNet-5 2**	**Other Arch**
Abbott	0.9450	0.9025	0.9550
Service	0.8800	0.9150	0.8800
Voros	0.9308	0.9110	0.9374

TABLE III: Total Loss

Test Dataset	Model Name **LeNet-5 1**	**LeNet-5 2**	**Other Arch**
Abbott	0.1365	0.3154	0.2302
Service	0.3036	0.3194	0.3546
Voros	0.1859	0.2979	0.2701

On all three datasets provided by each author, the different models seem to be performing well, with the minimum accuracy showing being only 88%.

VI. Future Improvements

There is still a lot of room for improvement when it comes to tackling the problem of facial recognition. Different deep learning architectures [3], signal processing techniques, and classical machine learning approaches are being implemented on the daily to continually improve the robustness, accuracy, and training time of these models.

A. New Architecture

Our group decided to get inspiration from [4] to try a model with the following repeating convolutional steps

TABLE IV: Model Training Metrics

Layer (type)
Conv2D
BatchNorm
Activation
MaxPooling
ZeroPadding
∧∧ **x3** ∧∧
Flatten
Dense
Dropout
Dense
Dropout
Dense

This model did not take a long time to run, as our training images are only $20 \times 20 \times 3$. However, the trade-off we experienced with this different architecture was a gain in accuracy with an increase in model size. The model had over ∼8 million parameters (compared to the ∼32,000 of LeNet), but experienced a 98.29% training accuracy and 96.35% validation accuracy (4.70% and -0.53% increase respectively from first LeNet-5 model) after only 15 epochs of training. The training history is seen in Fig. 4

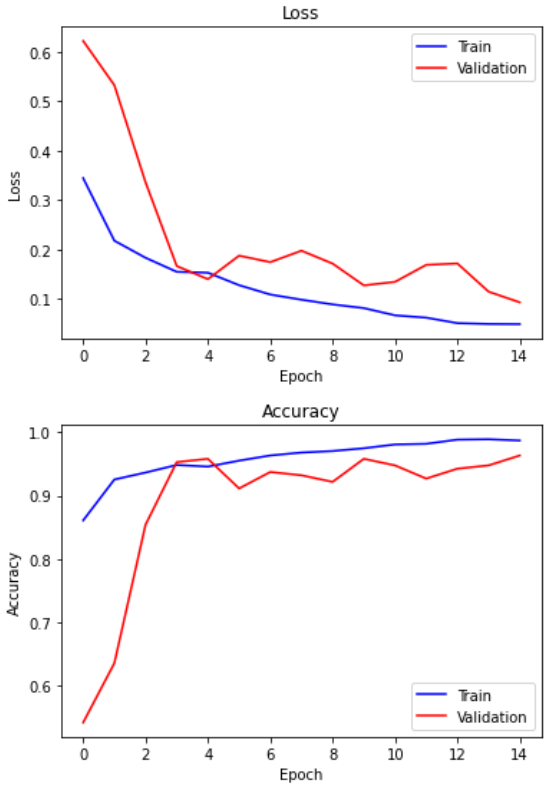


Fig. 4: Other architecture trained on data

It was observed that an increase in the number of parameters results in a more consistent, less sporadic curve for both training and validation accuracy.

VII. Appendix

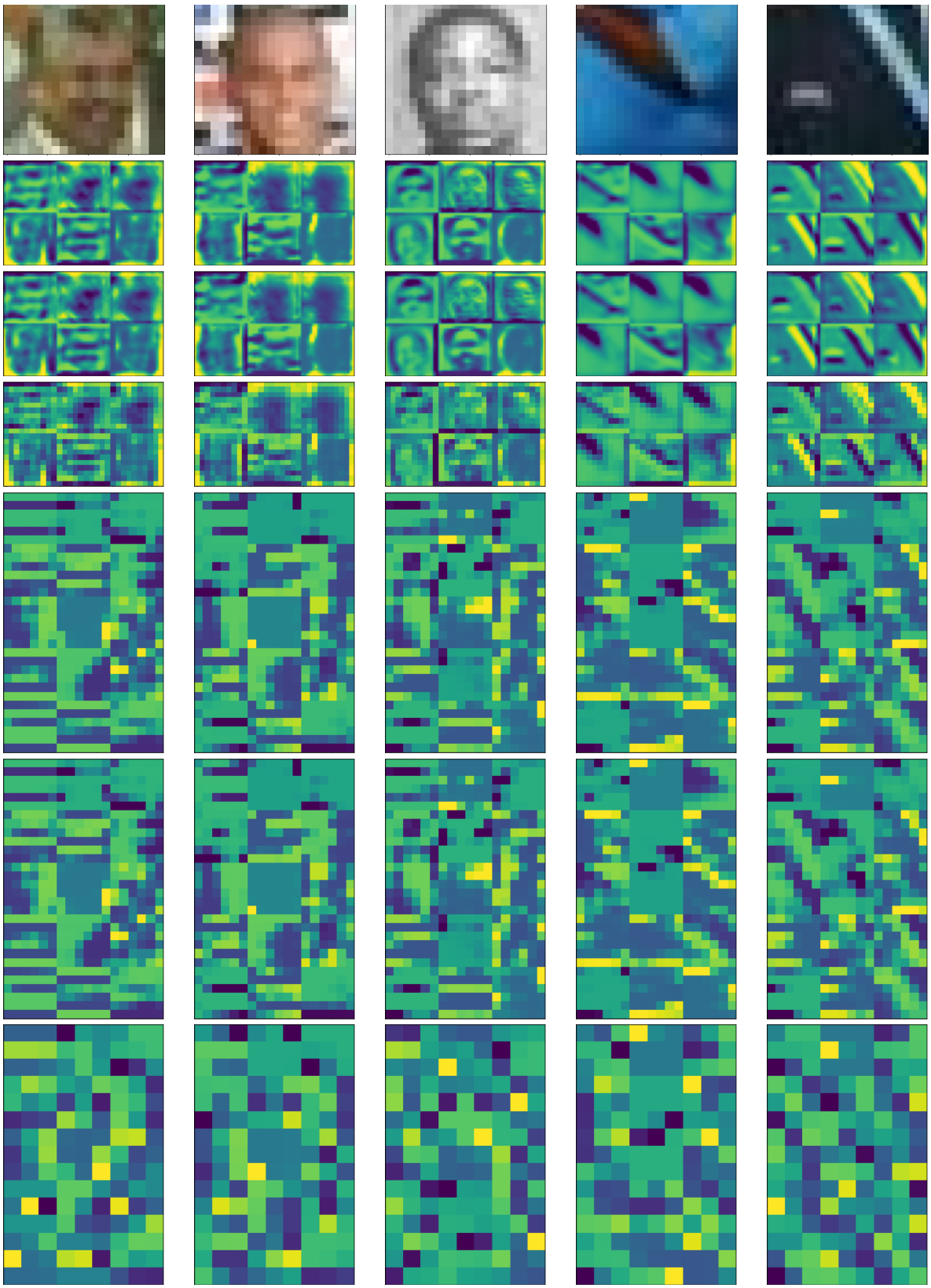
```
accuracy: 0.850760281085968, lr: 0.00020591547404285354, reg: 0.0016669775422667897, (1 / 100)
accuracy: 0.7139102816581726, lr: 2.058321131158766e-05, reg: 5969.084580555858, (2 / 100)
accuracy: 0.7458231449127197, lr: 3.895542757340805e-05, reg: 805.9428791419493, (3 / 100)
accuracy: 0.850760281085968, lr: 0.00015030739457648273, reg: 0.0007748409433049611, (6 / 100)
accuracy: 0.5154871344566345, lr: 1.12899425061332e-06, reg: 520.202890252447, (7 / 100)
accuracy: 0.8747888207435608, lr: 0.0008388833082081885, reg: 0.0008646020815512633, (21 / 100)
accuracy: 0.6767411231994629, lr: 7.184416204080309e-06, reg: 7156.758620143875, (22 / 100)
accuracy: 0.5671109557151794, lr: 0.0005460997953867529, reg: 2.1932297193009305, (23 / 100)
accuracy: 0.875164270401001, lr: 0.0005206196984289869, reg: 9.409938909036257e-05, (32 / 100)
accuracy: 0.6159188747406006, lr: 3.489969393091781e-06, reg: 445.26962045078164, (33 / 100)
accuracy: 0.8040172457695007, lr: 0.00018431006198261528, reg: 0.1205441065576137, (46 / 100)
accuracy: 0.8124647736549377, lr: 6.87044311772769e-05, reg: 35.05278300815377, (47 / 100)
accuracy: 0.6513985395431519, lr: 6.150787658381988e-06, reg: 0.0013207494500973747, (48 / 100)
accuracy: 0.8137788772583008, lr: 0.00010898959419726599, reg: 79.67665862829855, (49 / 100)
accuracy: 0.5275014042854309, lr: 1.6653228605581713e-06, reg: 4.386574960764426, (50 / 100)
accuracy: 0.46799322962760925, lr: 1.2014165310064732e-06, reg: 0.5049593215976883, (51 / 100)
accuracy: 0.45954570174217224, lr: 5.565979164381243e-06, reg: 0.07677390386133537, (52 / 100)
accuracy: 0.8267317414283752, lr: 3.156508077886206e-05, reg: 0.00011056021595854338, (53 / 100)
accuracy: 0.5676741003990173, lr: 0.0007089819968062401, reg: 90.35336799074332, (54 / 100)
accuracy: 0.8764783143997192, lr: 0.0008894211052917183, reg: 0.0012911771624703559, (55 / 100)
accuracy: 0.5098554491996765, lr: 1.1313123016827506e-06, reg: 2.59536567088212, (92 / 100)
accuracy: 0.8552656173706055, lr: 0.0002210919234675175, reg: 3.156947992629758e-05, (93 / 100)
accuracy: 0.8708466291427612, lr: 0.00034929842011266536, reg: 4.9079873228156995e-05, (94 / 100)
accuracy: 0.683311402797699, lr: 1.5470898901649485e-05, reg: 386.0851656997398, (95 / 100)
```

Fig. 5: Summarized Coarse Search Results

```
accuracy: 0.8790077567100525, lr: 0.0009886179068713233, reg: 4.494527863178021e-05, (1 / 100)
accuracy: 0.8731015920639038, lr: 0.0005467004059698439, reg: 0.0001480734781284659, (2 / 100)
accuracy: 0.8629767298698425, lr: 0.0002865027043933l7, reg: 1.3037531320685034e-05, (3 / 100)
accuracy: 0.849139392375946, lr: 0.00013208329891388305, reg: 0.000729415172784374, (4 / 100)
accuracy: 0.8779952526092529, lr: 0.0006894971877796719, reg: 0.0002079798410894499, (9 / 100)
accuracy: 0.8715828657150269, lr: 0.0009784754023105804, reg: 0.00021869700746689818, (10 / 100)
accuracy: 0.8801890015602112, lr: 0.0009500962470569233, reg: 5.0705420934477714e-05, (11 / 100)
accuracy: 0.8582517504692078, lr: 0.00014052802797730941, reg: 1.2039689099517143e-05, (65 / 100)
accuracy: 0.8639891743659973, lr: 0.00035158174031996837, reg: 0.00037287899988668494, (66 / 100)
accuracy: 0.8817077279090881, lr: 0.0009629125543273571, reg: 0.0001674783860941926s, (67 / 100)
accuracy: 0.8758015632629395, lr: 0.0009282918460241473, reg: 0.0008391950470831409, (68 / 100)
accuracy: 0.8518393635749817, lr: 0.0001654435464968225, reg: 1.670363480339818e-05, (93 / 100)
accuracy: 0.8450894355773926, lr: 0.00018050744152642894, reg: 0.0008117813555622789, (94 / 100)
accuracy: 0.8803577423095703, lr: 0.0007287962848401o5, reg: 0.00036980086551104824, (95 / 100)
accuracy: 0.8523455858230591, lr: 0.00015510031570618117, reg: 4.2628851755375506e-05, (96 / 100)
```

Fig. 6: Summarized Fine Search Results

TABLE V: Three positive and two negative image being passed through the LeNet-5 model



Face evaluation

0.9985031 0.9994142 0.9529020 0.0003845 0.0003776

REFERENCES

[1] S. Prince, *Computer Vision: Models Learning and Inference*. Cambridge University Press, 2012.

[2] G. Pierobon, "Visualizing intermediate activation in convolutional neural networks with keras," 2018.

[3] S. Xie, A. Kirillov, R. Girshick, and K. He, "Exploring randomly wired neural networks for image recognition," 2019.

[4] X. Bracquart, "Face recognition with keras," 2020.