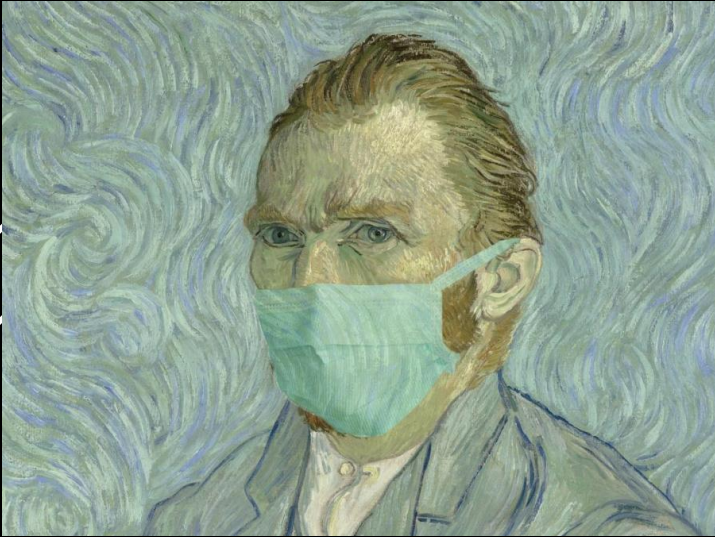


Are you wearing the mask?

Sinan Gültekin

095725

Neural Networks



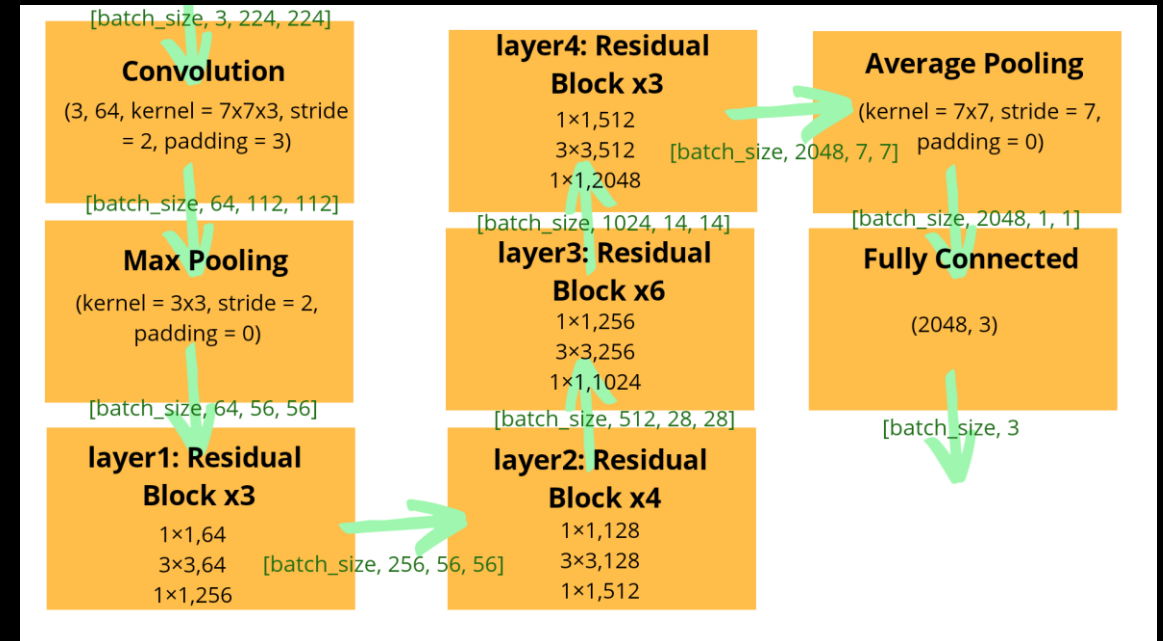
Outline

- ▽Brief Introduction
- ▽Directory Structure
- ▽Explanation of Project
- ▽Usage Examples
- ▽Discussion of Results

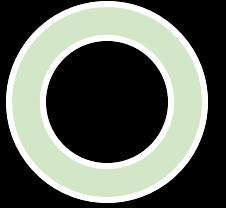
Brief Introduction: ResNet50

In this project, 3 main implementations of CNN will be presented. 2 of them are as followed ResNet50 networks:

1. ResNet50 – not_retrain: Transfer learning is performed from ResNet50, and only last layer is changed with respect to dataset output classes.
 - The networks parameters are frozen.
2. ResNet50 – retrain: Transfer learning is performed with fine-tuning of weights and biases of last 4 layers(layer3, layer4, avgpool, fc which can be seen on the figure on the right) original pre-trained ResNet50.
 - The network parameters of last 4 layers are unfreezed.

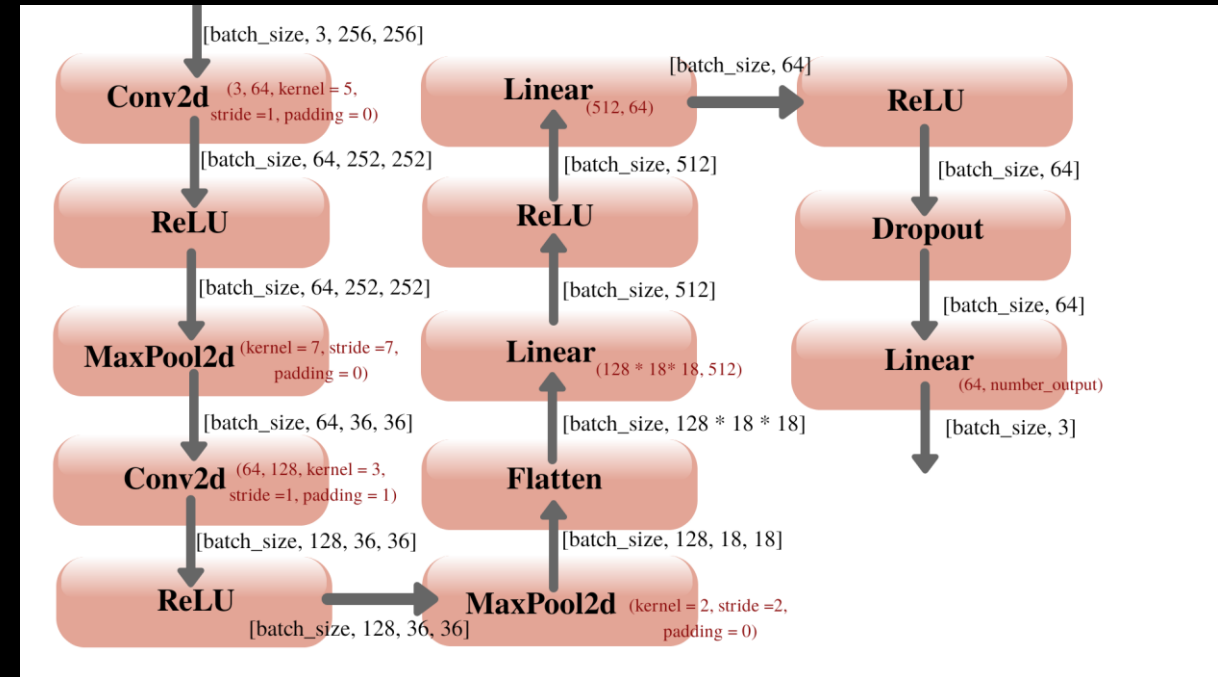


Brief Introduction: BasicCNN



3. BasicCNN: CNN from scratch is implemented. The figure on the right shows the structure of BasicCNN .

- The main body of the BasicCNN consists of convolutional layers, ReLU activation layers and max pooling layers.
- Flatten operation is applied to change the dimensions.
- In order to get more convenient results 2 linear and ReLU operations are applied sequentially.
- After that, dropout is used to get better results.
- Finally, with linear layer, softmax activation function is used to obtain output classes.



Brief Introduction: Dataset



The dataset of the project cannot be shared on publicly due to privacy issues. The dataset consists of 3 classes. Each class and its number of data follow as:

- Mask: 343
- No_mask: 304
- Wrong_mask: 400

The dataset is collected from friends and family members. However, the dataset structure is showed on the directory structure part.

In order to evaluate the performance of third(wrong_mask) class, several test runs completed with and without third class.

Due to the simplicity for saving the results of training and classification results, there are two different dataset directories for 2 and 3 classes cases.


```

.
├── dataset_2_classes
│   ├── labeled
│   │   ├── mask
│   │   └── no_mask
│   └── unlabeled
├── dataset_3_classes
│   ├── labeled
│   │   ├── mask
│   │   ├── no_mask
│   │   └── wrong_mask
│   └── unlabeled
├── demo_gifs
│   ├── ResNet-not_retrain-3classes-64-60-0.00100000.gif
│   ├── ResNet-retrain-2classes-64-60-0.00010000.gif
│   └── ResNet-retrain-3classes-64-60-0.00010000.gif
├── images
│   ├── BasicCNN.png
│   └── ResNet50.png
├── models
│   ├── BasicCNN-not_retrain-2classes-64-60-0.00010000.pth
│   ├── BasicCNN-not_retrain-2classes-64-60-0.00100000.pth
│   ├── BasicCNN-not_retrain-3classes-64-60-0.00010000.pth
│   ├── BasicCNN-not_retrain-3classes-64-60-0.00100000.pth
│   ├── ResNet-not_retrain-2classes-64-60-0.00010000.pth
│   ├── ResNet-not_retrain-2classes-64-60-0.00100000.pth
│   ├── ResNet-not_retrain-3classes-64-60-0.00010000.pth
│   ├── ResNet-not_retrain-3classes-64-60-0.00100000.pth
│   ├── ResNet-retrain-2classes-64-60-0.00010000.pth
│   ├── ResNet-retrain-2classes-64-60-0.00100000.pth
│   ├── ResNet-retrain-3classes-64-60-0.00010000.pth
│   └── ResNet-retrain-3classes-64-60-0.00100000.pth
├── packages
│   ├── __init__.py
│   ├── dataset.py
│   ├── network.py
│   └── utils.py
├── results
│   ├── BasicCNN-not_retrain-2classes-64-60-0.00010000.png
│   ├── BasicCNN-not_retrain-2classes-64-60-0.00100000.png
│   ├── BasicCNN-not_retrain-3classes-64-60-0.00010000.png
│   ├── BasicCNN-not_retrain-3classes-64-60-0.00100000.png
│   ├── ResNet-not_retrain-2classes-64-60-0.00010000.png
│   ├── ResNet-not_retrain-2classes-64-60-0.00100000.png
│   ├── ResNet-not_retrain-3classes-64-60-0.00010000.png
│   ├── ResNet-not_retrain-3classes-64-60-0.00100000.png
│   ├── ResNet-retrain-2classes-64-60-0.00010000.png
│   ├── ResNet-retrain-2classes-64-60-0.00100000.png
│   ├── ResNet-retrain-3classes-64-60-0.00010000.png
│   └── ResNet-retrain-3classes-64-60-0.00100000.png
├── __init__.py
├── face_mask_detector.py
├── haarcascade_frontalface_default.xml
├── LICENSE
├── README.md
├── requirements.txt
└── Terminal Command Run Examples.txt

```

Directory Structure

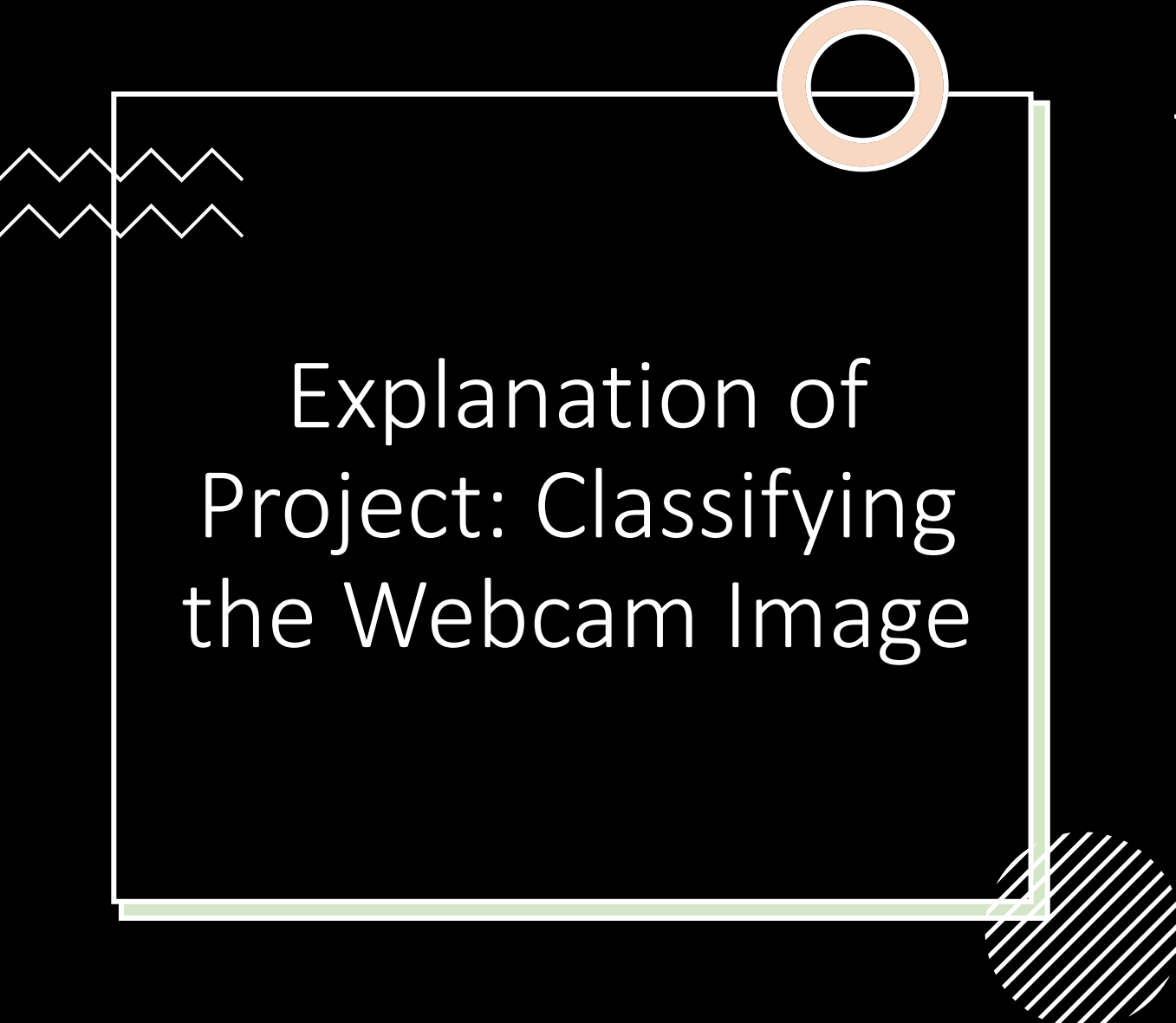


Explanation of Project: Training the Network

This part is responsible for

- Getting the dataset
 - The getting dataset is mostly computed with [packages.dataset](#). Inside this package, one can see the `packages.dataset.Dataset()` class, which is inherited from [torch.utils.data.Dataset](#), additionally `__len__()` and `__getitem__()` methods is overwritten on the torch module.
- Initializing the CNN models and training the 2 different dataset option with 3 different CNN models.
 - Initialization of CNN models is computed by [packages.network](#). Inside this package, there exists `packages.network.CNNClassifier()` class, that is highly inherited from [torch.nn.Module](#).
 - Data preprocessing is applied includes random rotation (+/- 20 degree), random crop scale (0.8/1.2), random resize ration (3/4 and 4/3) and finally random horizontal flip (0.5 probability).
 - Cross Entropy is used for loss function.
 - ADAM is applied for optimizer.
 - For making decisions, argmax is performed.

For any detailed information about classes and methods, please visit my [Github](#) page.



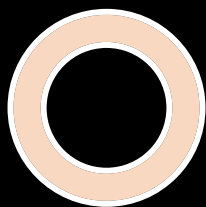
Explanation of Project: Classifying the Webcam Image

This step is responsible for

- The classification process on the input webcam image will be performed.
 - Computing classification is based on [packages.utils](#).
- Capturing webcam input will be destroyed when ESC key is pressed.



Usage Examples



Training:

To see help in terminal:

```
$ python3 face_mask_detector.py train -h
```

Positional argument (dataset_path) must be given to compile code. For optional ones, code can perform with default values as well.

One example usage with ResNet50 in retrain mode:

```
$ python3 face_mask_detector.py train  
./dataset3_classes/labeled/ --backbone ResNet --  
resnet_retrain_mode retrain --split_data [0.70, 0.20,  
0.10]
```

Evaluating:

To see help in terminal:

```
$ python3 face_mask_detector.py evaluate -h
```

Positional arguments (network_path and dataset_path) must be given to compile code. For optional ones, code can perform with default values as well.

One example usage with previously saved ResNet50 :

```
$ python3 face_mask_detector.py evaluate ./models/ResNet-  
not_retrain-64-3-0.001.pth ./dataset_3_classes/labeled/ --  
resnet_retrain_mode not_retrain --shuffle False
```

Classify:

To see help in terminal:

```
$ python3 face_mask_detector.py classify -h
```

Positional argument (network_path) must be given to compile code. For optional ones, code can perform with default values as well.

One example usage with ResNet50 with retrained model:

```
$ python3 face_mask_detector.py classify models/ResNet-retrain-  
2classes-64-60-0.00010000.pth
```



Discussions of Results

2 main outcome can be seen:

- 1. First observation one can make is that, training the networks ResNet50 and BasicCNN with 2 classes dataset gives more robust results.**
 - **Even though**, 2 classes dataset is quite small, 647 images in total, networks can perform descent performance.
 - Since BasicCNN is a network, will be trained from scratch, 647 data is not enough to perform similar to ResNet50.
 - In the case of 3 classes dataset, 1047 images in total, the top two performances were reached by not retrained and retrained ResNet50 with different learning rates. When retrain is needed, going lower learning rates may be better option to train network with current dataset.
- 2. The second observation one can come up is that, validation accuracy of ResNet50 can be greater than training accuracy.**
 - The possible answer for this problem is using of dropouts.
 - Dropout technique is only used during training, it is not used while evaluation is processing.
 - This results with getting stuck of validation accuracy since disabling neurons, some of the information about each data sample is disappeared, and as a result of that subsequent layers try to make predictions based on lost informations.
 - Training accuracy is lower because dropout method makes networks prediction harder to get correct predictions.
 - However, during the validation process network can use all neurons and their informations. Hence, network has all of its computational power to predict the class and it might perform better.

model	training acc.	validation acc.	test acc.
ResNet-not_retrain-2classes-64-60-0.00100000.pth	96.04	100.00	98.00
ResNet-not_retrain-2classes-64-60-0.00010000.pth	94.49	100.00	95.00
ResNet-retrain-2classes-64-60-0.00100000.pth	98.90	98.96	95.00
ResNet-retrain-2classes-64-60-0.00010000.pth	97.80	100.00	99.00
BasicCNN-not_retrain-2classes-64-60-0.00100000.pth	79.52	98.88	73.00
BasicCNN-not_retrain-2classes-64-60-0.00010000.pth	82.60	95.83	74.00
ResNet-not_retrain-3classes-64-60-0.00100000.pth	91.67	84.62	84.28
ResNet-not_retrain-3classes-64-60-0.00010000.pth	89.07	80.77	76.73
ResNet-retrain-3classes-64-60-0.00100000.pth	85.52	83.33	77.99
ResNet-retrain-3classes-64-60-0.00010000.pth	97.27	95.51	82.39
BasicCNN-not_retrain-3classes-64-60-0.00100000.pth	89.62	75.00	67.30
BasicCNN-not_retrain-3classes-64-60-0.00010000.pth	78.01	71.15	62.89

>Model Name Explanation:	ResNet -> backbone
	not_retrain -> resnet_retrain_mode
	2classes -> dataset classes
	64 -> batch size
	60 -> epochs
	0.00100000 -> learning rate
	.pth -> .file extension

Note: resnet_retrain_mode is not used with BasicCNN backbone. Parameter is just here for getting consistent model names.

Discussions of Results: Demos

