POLITECNICO
MILANO 1863

ARTIFICIAL NEURAL NETWORKS AND DEEP LEARNING

(Prof. Matteo Matteucci, Prof. Giacomo Boracchi, Prof. Francesco Lattari)

TEAM The Neurons Burners

First Assignment: Classification Problem on Mask Dataset

Luca Colasanti

Stefano D'Angelo

Enrico Pancher

Academic Year 2020-2021

The aim of the project is to train a neural network able to classify images depicting people in three different classes:

- Class 0: no person wears a mask;
- Class 1: all people wear a mask;
- Class 2: only some people wear a mask.

This document explains line of reasoning and the various steps that led to the production of the final model.

Data Manipulation:

The first step was to split the training dataset into two different ones: training and validation. This was made to obtain a comparable number of images for each class both in the training and in the validation sets. The split resulted in the following number of images (see notebooks for further information):

	Class 0	Class 1	Class 2	Tot # images
Training Dataset:	1417	1421	1362	4200
Validation Dataset:	483	476	455	1414

Data Augmentation:

To compensate for the low population of the dataset, data augmentation was applied to the training dataset. The following transformations were used:

- Horizontal flip;
- Brightness change.

All transformations that could cause "label pollution" in the original dataset have been avoided (i.e.: zoom, vertical or horizontal shift could cause removal of masked individuals in images with label 2).

Model Building:

1) Self-designed models:

Several procedures were adopted to produce the final model: the first approach was to build it from scratch. Many models were tested and are listed in Table 1. All models are made up of 5 convolutional blocks (convolutional layer with kernel size=(3,3), stride=(1,1), padding=same) and a classifier composed by two layers.

Models performances were tested by changing:

- number of filters for each block;
- number of neurons for each layer in the classifier;
- batch size;
- learning rate;

Many activation functions were tried (ReLU, ELU, LeakyReLU) while AdaDelta was the optimizer used for all these models. Early stopping technique is applied to prevent overfitting, many patience levels are used. Graphs obtained in TensorBoard and limited accuracy of the submitted uploads highlighted the necessity to try a new method based on transfer learning.

MODEL NAME	ACCURACY	(IMG_H,IMG_W)	BATCH SIZE	# FILTERS	DEPTH	KERNEL SIZE	ACTIV.FUNCTION	#DENSE_LAYERs, #NEURONSperLAYER	L. RATE	PATIENCE(in E.S)	#EPOCHS_DONE
Model_09_Nov13_17-33-55	0.64888	(256, 256)	8	32	5	(3, 3)	ReLU	2, 2048, 1024	1.00E-02	20	153 / 200
Model_08_Nov13_09-34-33	0.62	(256, 256)	8	32	5	(3, 3)	ReLU	2, 2048, 1024	5.00E-03	20	100 / 100
Model_07_Nov13_01-05-53	0.57333	(256, 256)	8	64	5	(3, 3)	ReLU + swish	2, 512, 256	5.00E-03	20	70 / 100
Model_02_Nov15_12-33-03	0.54888	(256, 256)	40	32	5	(3, 3)	ReLU	2, 1024, 512	1.00E-02	10	88/100
Model_06_Nov12_19-31-58	0.52666	(256, 256)	8	8	6	(3, 3)	LeakyReLU	2, 512, 256	5.00E-03	20	90 / 100
Model_04_Nov12_17-28-04	0.49555	(256, 256)	8	8	5	(3, 3)	LeakyReLU	2, 512, 256	1.00E-02	10	18 / 100
Model_03_Nov12_15-36-03	0.42222	(256, 256)	8	8	5	(3, 3)	LeakyReLU	2, 512, 256	1.00E-03	10	100 / 100
Model_02_Nov12_14-54-34	0.40222	(256, 256)	8	8	5	(3, 3)	ELU	2, 512, 256	1.00E-03	7	36 / 100
Model_01_Nov12_10-20-51	0.37777	(256, 256)	8	8	5	(3, 3)	ELU	2, 512, 256	1.00E-04	7	70 / 70

Table 1: Self designed models

2) Transfer Learning models:

Transfer learning is applied with different pre-trained models: VGG16, InceptionV3 and VGG19. Weights initialization is done with ones derived from Imagenet. Original classifier is removed and replaced by another one with variable structure. Different learning rates have been applied and Adam optimizer is used for some models in this trial session.

MODEL NAME	ACCURACY	(IMG_H,IMG_W)	BATCH SIZE	CNN	WEIGHTS	ACTIV.FUNCTION	#DENSE_LAYERs, #NEURONS	L. RATE	OPTIMIZER	PATIENCE(in E.S.)	#EPOCHS_DONE
Model_13_Nov14_13-09-32	0.60222	(224, 224)	9	VGG19	imagenet	relu	2, 512, 256	5.00E-03	Adadelta	7	22 / 100
Transfer_Learning-InceptionV3-Project2020	0.18666	(224, 224)	32	InceptionV3	imagenet	relu	2, 512 +Dropout(0.5)	1.00E-04	Adam	0	28/80
Model 11 Nov14 01-03-35	0.61333	(256, 256)	8	VGG16	imagenet	relu	2, 512, 256	1.00E-03	Adadelta	15	29 / 100

Table 2: Transfer Learning Models

3) Fine tuning models:

To further improve performances and to ensure greater adherence to the dataset in the extraction of more abstract features, fine tuning has been applied by training only the weights of the last layers. In some cases, additional global pooling layers or convolutional layers have been added to the CNN. Different numbers of frozen layers or classifier structures have been applied. Adam optimizer was used for all trials.

NOME MODELLO	ACCURACY	(IMG_H,IMG_W)	BATCH SIZE	CNN	WEIGHTS	ACTIV.FUNCTION	FROZEN LAYERS	#DENSE_LAYERs, #NEURONS	L.RATE	PATIENCE(in E.S.)	#EPOCHS_DONE
Model_16_Nov17_00-17-39	0.86888	(224, 224)	24	VGG19 + (4conv)	imagenet	relu	[:15]	3, 1024 + batch, 512 + batch, 256 + batch	1.00E-02	7	8 / 100
Model_27 Nov20_20-46-11	0.83555	(224, 224)	24	VGG19 + globalPooling	imagenet	relu	[:18]	3, 512 + batch, 256 + batch, 128 + batch	1.00E-03	20	48/100
Model_17_Nov15_14-28-08	0.80444	(224, 224)	10	VGG16 + (4 conv)	imagenet	relu	[:15]	2, 512 + batch, 256 + batch	1.00E-02	5	14 / 100
Model_12_Nov14_09-41-34	0.73777	(224, 224)	9	VGG19	imagenet	relu	[:13]	1,512	1.00E-03	7	14/100
Model_10_Nov13_21-46-13	0.70888	(256, 256)	8	VGG16	imagenet	relu	[:15]	1, 512	1.00E-03	3	20 / 20

Table 3: Fine tuning models

4) Fully fine tuned models:

The strategy used to produce the final model was the same as in point 3) but without freezing any layer. Table 4 gives an overview of the tested models.

MODEL NAME	ACCURACY	(IMG_H,IMG_W)	BATCH SIZE	CNN	WEIGHTS	ACTIV.FUNC	#DENSE_LAYERs, #NEURONSperLAYER	L. RATE	OPTIMIZER	PATIENCE(in E.S.)	#EPOCHS_DONE
Model_55_Nov20_10-36-0	**0.95333**	**(224, 224)**	**24**	**VGG19 + [(1conv + 1bn + 1 relu)*4+Max pool.] x2**	**@Model54-epoch27**	**relu**	**3, 1024 + batch, 512 + batch, 256 + batch**	**1.00E-04**	**Adam**	**20**	**31/ 100**
Model_51_Nov20_11-44-50	0.93555	(224, 224)	24	VGG19 + [(1conv + 1bn + 1 relu)*4+Max pool.] x2	imagenet	relu	3, 1024 + batch, 512 + batch, 256 + batch	1.00E-04	Adam	20	33/ 100
Model_54_Nov20_08-36-1	**0.91555**	**(224, 224)**	**24**	**VGG19 + [(1conv + 1bn + 1 relu)*4+Max pool.] x2**	**imagenet**	**relu**	**3, 1024 + batch, 512 + batch, 256 + batch**	**1.00E-04**	**Adam**	**20**	**43/ 100**
Model_28_Nov19_15-45-41	0.90666	(224, 224)	10	VGG19 + (4 conv + batch)	imagenet	relu	3, 512 + batch, 256 + batch, 128 + batch	1.00E-04	Adam	10	38 / 100
Model_53_Nov20_00-16-54	0.89333	(224, 224)	24	VGG19 + (4 conv + batch)	imagenet	relu	3, 1024 + batch, 512 + batch, 256+ batch	1.00E-04	Adam	20	43/ 100
Model_21_Nov17_15-19-16	0.88444	(224, 224)	24	VGG19 + (4conv + bn + 4 conv + bn)	imagenet	relu	3, 1024 + batch, 512 + batch, 256 + batch	1.00E-02	Adadelta	15	33/ 100
Model_30_Nov17_20-50-18	0.88666	(224, 224)	22	VGG19 + (4conv + bn + 4 conv + bn)	imagenet	relu	3, 1024+ batch, 512+ batch, 256+ batch	1.00E-02	Adadelta	15	24/ 100
Model_17_Nov17_01-15-18	0.88	(224, 224)	24	VGG19 + (4conv + bn)	imagenet	relu	3, 1024 + batch, 512 + batch, 256 + batch	1.00E-02	Adadelta	15	29/ 100
Model_18_Nov17_11-36-54	0.87111	(224, 224)	24	VGG19 + (4conv + batch normalization)	imagenet	relu	3, 1024 + batch, 512 + batch, 256 + batch	1.00E-02	Adadelta	15	19/ 100
Model_21_Nov16_11-05-33	0.86222	(224, 224)	10	VGG19 + (3 conv + batch)	imagenet	relu	3, 512 + batch, 256 + batch, 128 + batch	1.00E-02	Adadelta	7	29 / 100
Model_20_Nov16_01-24-51	0.87333	(224, 224)	10	VGG19 + (1 conv + batch)	imagenet	relu	3, 512 + batch, 256 + batch, 128 + batch	1.00E-02	Adadelta	7	32 / 100

Table 4: Full fine tuning models, highlighted models are the one used to produce final delivered model

The delivered model was obtained by retraining a network structured as follows:

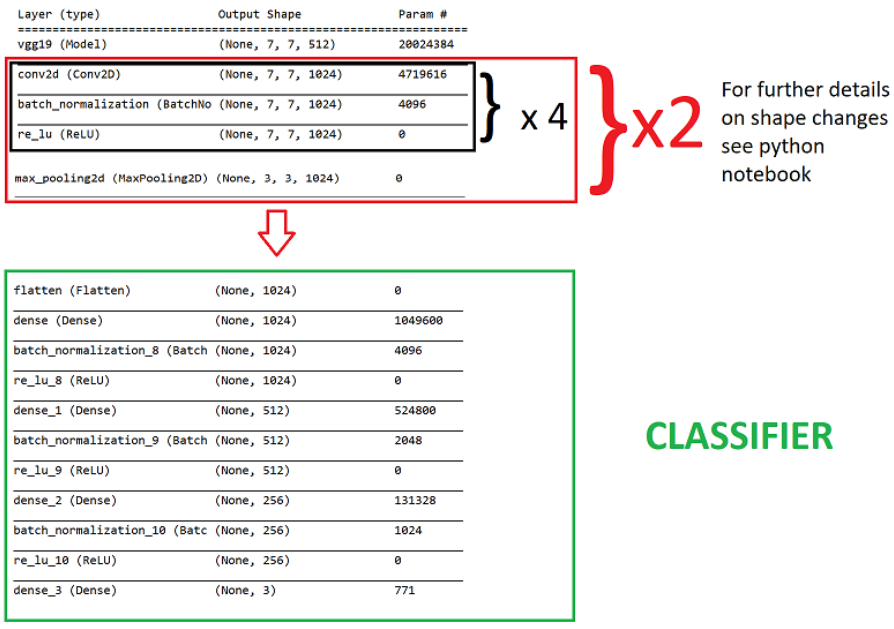

Fig. 1: Final model structure

Batch Normalization is present both in convolutional blocks and in classifier because it has shown to:

- Improve gradient flow through the network;
- Allow higher learning rates;
- Act as a form of regularization, reducing the need for dropout.

```
SEED = 1234 --> used to guarantee experiments  reproducibility
rescale(applied on all pixels of all images) = 1./255
img_h = 224
img_w = 224
bs = 24
shuffle (of images in training dataset)=True
loss function = tf.keras.losses.CategoricalCrossentropy()
learning rate = 1e-4
optimizer = Adam
EarlyStopping (monitor='val_loss', patience=20)
# epochs = 100
```

the following formula was used to calculate the maximum batch size

$$MaxBatchSize = \frac{AvailableGPUmemoryBytes}{4 * (SizeOfTensors + TrainableParameters)} = 24$$

where: *size of tensors = img_h x img_w x #channels* , and *#channels = 3* in case of **RGB**
Source: https://stackoverflow.com/questions/46654424/how-to-calculate-optimal-batch-size

Fig. 1: Main hyperparameters and optimizer parameters

Model Training:

Final delivered model is called Model 55. To obtain it, a model (Model 54) with the same structure and hyperparameters has been trained. To ensure that the best weights combination was obtained as well as to avoid blocking the update of weights at a local minimum in the loss function it has been decided to use a patience of 20 epoch for the early stopping. The followed steps were:

- The model structured as in Fig.1 has been trained with initial weights taken from Imagenet. As an output of the procedure Model 54 has been obtained, stopped from early stopping at epoch 22;
- Epochs of Model 54 were analysed to detect the epoch with highest validation accuracy. This results to be epoch n.27 (val_accuracy= 0.924);
- Weights at epoch 27 have been extracted and used as new initialization weights for the same model structure.

Then, the model has been retrained with these new initial weights and the training session produced our final model (early stopping at epoch 10, val_loss= 0.2377, val_accuracy=0.924, test accuracy=0.95333). Validation accuracy of this model is in a satisfying range (from 0.835 to 0.986).

See next page for TensorBoard graphs.

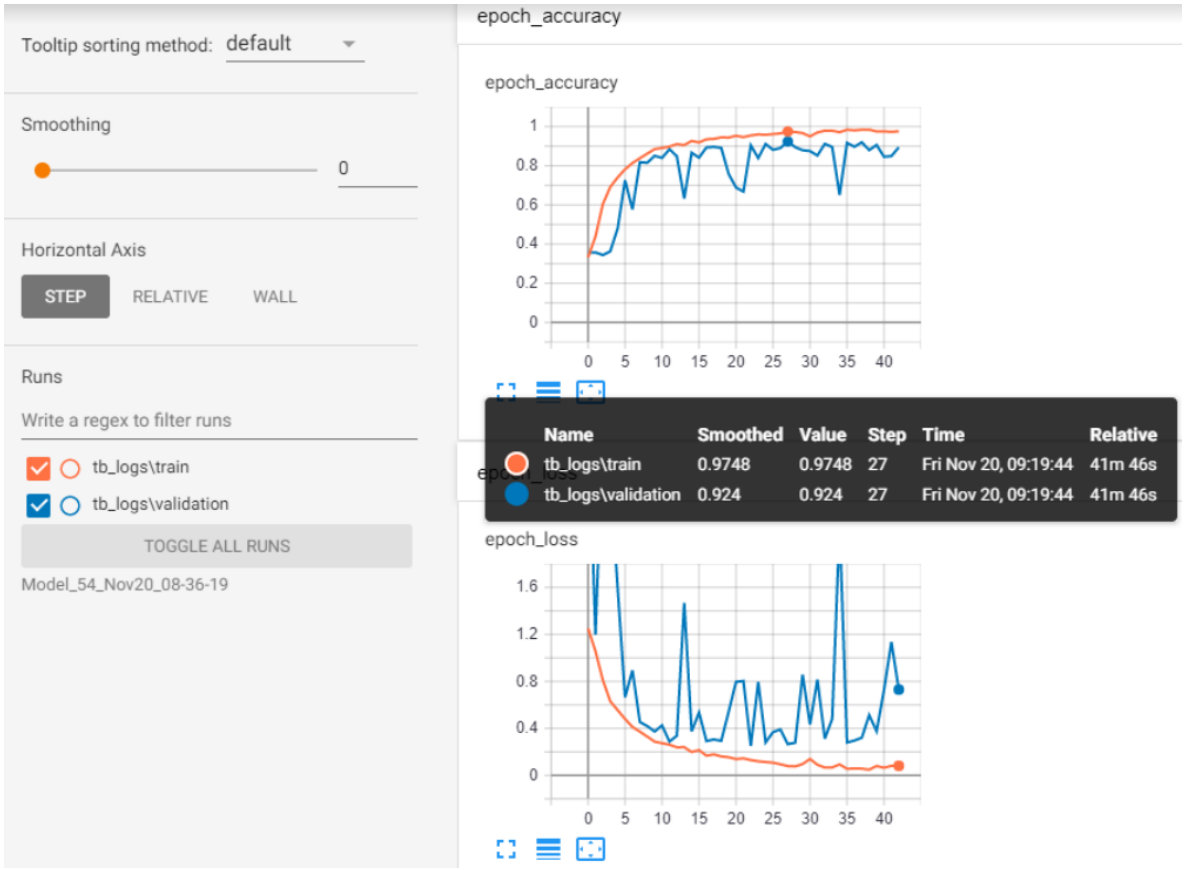

Figure 2: Tensorboard graphs of Model 54: Pointer highlights Epoch 27 for which max Valid. Accuracy is obtained

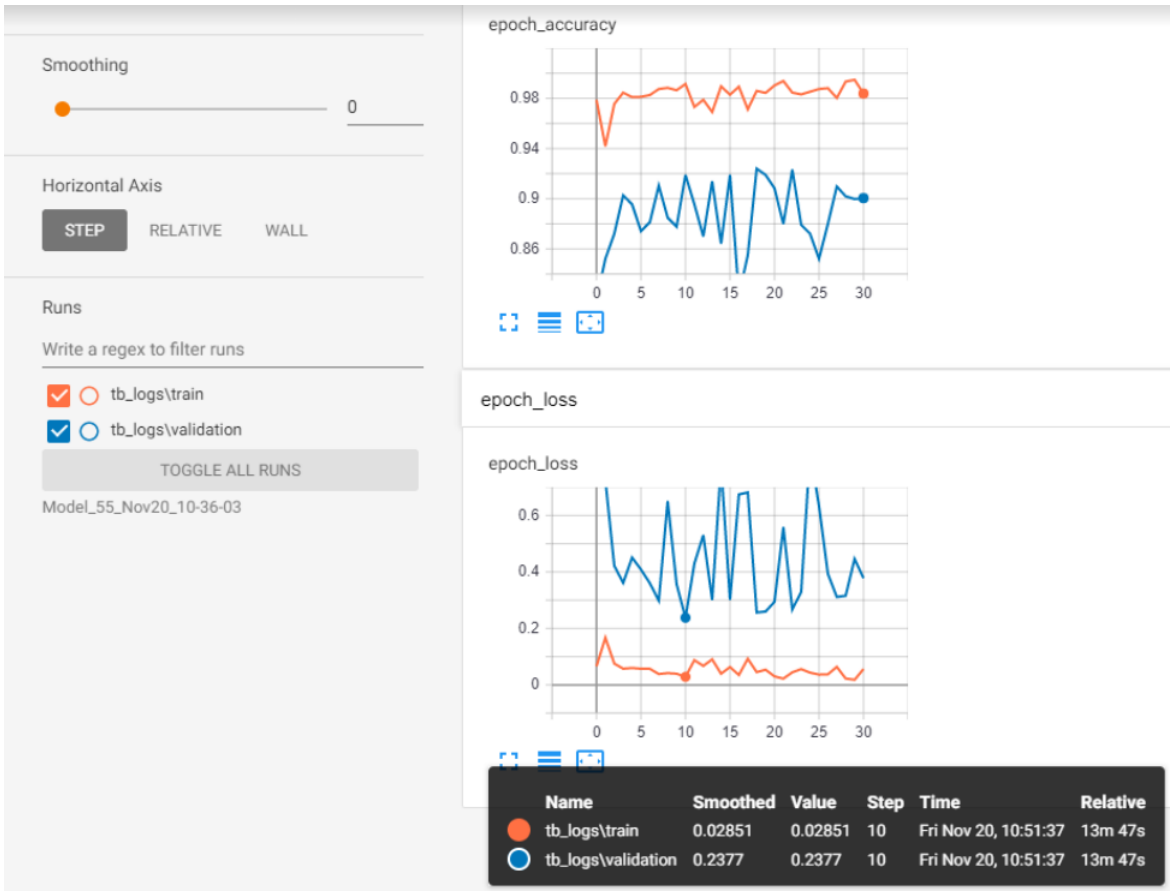

Figure 3: Tensorboard graph of Final Model (Model 55): Pointer highlights epoch 10 where early stopping stops the training