Deep Learning KU (708.220) WS22

Assignment 3: Convolutional Neural Networks

David Mihola
12211951

December 21st, 2022

Contents

a) Model architecture selection

I chose two architecture philosophies and tried three different variations of each, i.e. six different models in total. The first architecture is remotely inspired by the Darknet-53 described in [1]. Although the Darknet-53 is designed for object detection, I thought, it could be also used for classification after applying some modifications. The second architecture is inspired by the VGG Net described in [2].
The idea of the first architecture is to have a convolutional layer with kernel size 3x3 followed by a pooling layer until the image width and height are not 1. Afterwards there is one or more convolutional layers with kernels of size 1x1, which compress the N channels obtained by the 3x3 convolution to 10 channels used for classification.
The second architecture consists of two or three VGG blocks followed by two or three fully connected layers. Each VGG block consists of two convolutional layers with 3x3 kernel size and a max pooling layer.
The training regarding the validation accuracy is summarized in table 1. All the models where trained with the default Adam optimizer settings and the default 0.01 learning rate for simplicity. I used early stopping to maximize the validation accuracy with patience for three epochs. The best result for each model and the overall best result are highlighted. The row header specifies the training epoch and the column header describes the model, where:

- `DN` stands for a Darknet-53 inspired model,
- `SP` is a shortcut for stride pooling layers, where pooling is done by additional 3x3 convolution layers with stride 2 and the same number of channels as the preceding convolutional layer,
- `MP` is a shortcut for max pooling layers with stride 2,
- `VGG` stands for a VGG net inspired model,
- `X-Y` describe the number of channels in the first convolutional layer and the last convolutional layer respectively, in each layer in case of DN and in each block in case of VGG the number of channels is doubled,
- `X, Y, ...` describe the sizes of fully connected layers including the output layer.

All models use the ReLu activation function for the hidden layers.
The average performance of the models is slightly above 70 %. The worst performing model is the first model in the table, although having more trainable parameters than all the other models apart from the third. This suggests, that the stride pooling simply does not work. On the other hand, the best performing model is the last one in the table, which has only more trainable parameters than the second and fifth model. This supports the fact, that having more trainable parameters is not necessarily better and that the arrangement of the trainable parameters in to various layers has a large impact on the models performance.

b) Regularization

I tried various different regularization techniques, which include dropout, l_1-regularization, l_2-regularization, $l_1 l_2$-regularization and also batch normalization. I ran a simple grid search with three different parameters for each regularizer or a combination of regularizers. I again used early stopping to maximize the validation accuracy with patience for three epochs. The results are presented in tables 2 and 3. The column header remains unchanged. The row header contains the used regularizer or regularizers and its parameters. Each cell contains five values, which are respectively the epoch, in which the best validation accuracy was reached, the training loss, the validation loss, the training accuracy and the validation accuracy. Cells, where training was not successful, are colored in gray. The best result for each model is highlighted in blue and the best result for each regularization technique is highlighted in green. The best overall result is highlighted in red. The values in the cells in the left upper corner of the table 2 are repeating, as the Darknet-53 inspired models do not have any layers, where dropout can be applied.
The best performing regularization technique for each model improves its result regarding the validation accuracy. Some improvements are less significant, e.g. second and third model, and some are more significant, e.g. first, fourth and fifth model. It can be seen, that the combination of batch normalization and dropout gives the best performance for all models. Most of the regularizers with the regularization weight of 0.01 and 0.001 prevent the models to learn at all. This is very likely caused by the regularization weight being set too high, so large weights and kernel values are punished too much and the models are not capable of extracting any meaningful features, especially in case of the convolutional layers. When the regularization weight is set to 0.0001, the initialization of weights and kernel values has an impact on the ability of the models to be trained, as consistently different models are trained and not trained when rerunning the experiment multiple times with random initializations.

c) Final model architecture summary

The summary of the final model, i.e. `VGG - MP; 32-128; 256, 10` with batch normalization and 0.4 dropout layers, which performed the best in the previous two sections, is given in the table 4. The column header is self explanatory and the row header tells the layer number. The used shortcuts in the table should be interpreted as the following:

- `B` – batch size, the batch size is not strictly given by the model and can vary,
- `CONV` – convolutional layer,
- `BN` – batch normalization layer,
- `MP` – max pooling layer,
- `FT` – flattening layer,

- `DO` – dropout layer,
- `DENS` – dense layer.

I trained the final model using the default settings of the Adam optimizer with a fixed learning rate of 0.01. I used the categorical cross entropy loss function. I kept the default value for batch size, i.e. 32, and run the training for 13 epochs, which is the number of epochs with the best validation accuracy from the previous experiment.
I also tried to rerun the experiment from the previous section with Adam optimizer and scheduled learning rate by an exponential decay function, which was the best performing learning rate schedule in the last assignment. The parameters for the exponential decay scheduler were 0.01 initial learning rate, decay steps of 1500 and a decay rate of 0.9. Other training parameters remained the same. Nevertheless this modification of the learning rate did not yield any improvements, so I kept the fixed learning rate of 0.01.
The progress of training of the final model still with the validation data set is plotted on the figure 1. It can be seen, that the model starts to overfit, but thanks to the relatively high dropout rate, the overfitting is not large. The accuracy of the model trained with the validation data set on the test data set is **80.27 %**. Including the validation data set also for training of the model improved the accuracy on the test data set to **81.44 %**. The miss classified samples from the test data set can be seen in the plot of the confusion matrix in figure 2. It seems, that the model struggles more with classification of images with animals. In fact, the highest number of miss classifications is, when the model confuses a cat with a dog or vice versa. This is probably given by the fact, that the images are very low resolution and converted to gray scale. Cats and dogs might look very similar in this setting even to a human. Another observation, which can be made from the confusion matrix, is, that when there are miss classifications, they usually remain in the super-class of the image, i.e. an animal is miss classified as an another animal and a vehicle is miss classified as an another vehicle. There is one exception, when a bird is also often miss classified as an airplane, which is probably given by the settings, in which birds and airplanes are usually photographed.

d) Perturbed data set evaluation and model modifications

The accuracy of the unchanged final model on the perturbed data set is **63.58 %**, which is a decrease from the test data set by nearly 20 %. To increase the accuracy on the perturbed data set I carried out experiments described in the following paragraphs.
Firstly, I inspected the perturbed data set and I was able to identify two of the seemingly three types of perturbation. The first perturbation is done by applying a salt and pepper noise and the second by applying a gaussian noise. For the third perturbation, it seems, there is some change in colors. I tried augmenting the original images by changing the brightness and the contrast, but I was not able to obtain the same perturbation.
Secondly, I tried three modifications of the final model. The salt and pepper perturbation lead me to think, that an average pooling layers could perform better than the currently used max pooling layers,

as the influence of the perturbed pixels should be diminished by averaging them with surrounding correct pixels. The second modification is adding l_2-regularization to each convolutional layer, replacing dropout with l_2-regularization for the hidden fully connected layer and also adding batch normalization after this layer. The third modification differs from the second only by, that the dropout is kept with decreased rate to 0.25. With the second and third modification, I thought, that increasing the overall amount of regularization should help with generalization on the perturbed data set. I trained all of the three modified models with the same optimization settings described in the previous section. None of these modifications lead to any improvement on both the test data set and more importantly on the perturbed data set. In case of the first modification, the difference in performance between the unchanged final model is negligible, which is probably given by the modification being very small. In fact, I trained this model multiple times with random weight initializations and sometimes it performed slightly better on the perturbed test data set than the unchanged final model. Regarding the second modification, the worse results might be caused by the model starting to overfit given the high accuracy on the training data set. The third modification indicates, that combining weight regularization with dropout might not be optimal, as the weight regularizer punishes also weights, which were dropped out in a given training step. The results can be seen in the table 5.

Lastly, I tried making my own perturbed training data set by augmenting the current one. For each existing image in the data set, I added an image with salt and pepper noise with 0.0175 proportion of pixels replaced by noisy ones and an image with gaussian noise with variance of 0.005. I decided to ignore the third perturbation, as it is not as disrupting as the other two and kept the original images instead. I retrained the final model on the perturbed training data set with the same optimization settings described in the previous section. The accuracy of the model on the perturbed test data set increased almost to the accuracy on the test data set, while the accuracy on the test data set remained very high, as can be seen in the table 5. Although the salt and pepper noise and the gaussian noise are common for image augmentation, in reality, when not knowing the data at inference time at all, I could also choose some other augmentation techniques, such as some geometrical transformations or manipulations of the colors of the images. These augmentations would probably decrease the accuracy on this specific perturbed test data set. To demonstrate that, I created another perturbed training data set with rotated images up to 20 degrees and images with increased brightness by 0.15 and increased contrast by 0.1. I also left the original images in this data set. The improvement on the perturbed test set is still not negligible, as can be seen in the table 5, but interestingly the accuracy on the original test data set improved to the best value seen so far. This is most likely caused by the fact, that this training data set, although with augmented images, is still three times larger than the original one.

In conclusion, the best performance on both the test data set and the perturbed test data set was reached with the model and the optimization scheme described in the previous section. The best accuracy reached on the test data set is **82.91 %**, when the model is trained on a perturbed training data set by rotation and color distortions. The best accuracy reached on the perturbed test data set is **78.24 %**, when the model is trained on a perturbed data set by salt and pepper noise and gaussian noise.

Attachments

This section provides figures and tables with results of carried out experiments described in the previous sections.

	DN – SP 16-256 10	**DN – MP 16-256 128, 10**	**DN – MP 32-512 256, 10**	**VGG – MP 32-64 256, 128, 10**	**VGG – MP 16-64 128, 10**	**VGG – MP 32-128 256, 10**
1	37.84 %	47.25 %	51.21 %	58.09 %	49.19 %	54.91 %
2	50.85 %	59.96 %	63.60 %	66.20 %	60.60 %	64.63 %
3	55.49 %	62.76 %	64.63 %	69.24 %	65.05 %	68.91 %
4	60.37 %	65.16 %	69.75 %	69.41 %	68.99 %	70.94 %
5	62.03 %	66.33 %	69.43 %	70.74 %	69.83 %	72.08 %
6	63.94 %	68.44 %	69.59 %	71.01 %	71.05 %	72.75 %
7	64.51 %	68.71 %	69.80 %	70.30 %	71.58 %	**73.86** %
8	64.65 %	68.39 %	69.75 %	**71.06** %	71.82 %	73.15 %
9	64.73 %	**68.78** %	70.09 %	70.19 %	**72.13** %	72.07 %
10	**64.95** %	67.61 %	68.00 %	69.73 %	71.04 %	72.90 %
11	64.71 %	66.82 %	69.05 %	68.26 %	69.68 %	—
12	63.61 %	67.58 %	70.73 %	—	69.98 %	—
13	63.10 %	—	69.53 %	—	—	—
14	—	—	**71.42** %	—	—	—
15	—	—	69.31 %	—	—	—
16	—	—	70.18 %	—	—	—
17	—	—	69.12 %	—	—	—

Table 1: Validation set accuracies of different models during epochs of training

	DN – SP 16-256 10	**DN – MP 16-256 128, 10**	**DN – MP 32-512 256, 10**	**VGG – MP 32-64 256, 128, 10**	**VGG – MP 16-64 128, 10**	**VGG – MP 32-128 256, 10**
batch normalization, dropout 0.5	7 0.53 0.82 81.71 % **72.82 %**	6 0.54 0.83 80.93 % **73.09 %**	5 0.50 0.82 82.92 % **73.93 %**	23 0.60 0.64 79.96 % 78.84 %	16 0.62 0.64 78.86 % 78.45 %	17 0.36 0.64 87.59 % **80.69 %**
batch normalization, dropout 0.4	7 0.53 0.82 81.71 % **72.82 %**	6 0.54 0.83 80.93 % **73.09 %**	5 0.50 0.82 82.92 % **73.93 %**	19 0.49 0.64 83.57 % **78.93 %**	18 0.51 0.66 82.51 % 78.36 %	13 0.39 0.60 86.58 % **81.23 %**
batch normalization, dropout 0.3	7 0.53 0.82 81.71 % **72.82 %**	6 0.54 0.83 80.93 % **73.09 %**	5 0.50 0.82 82.92 % **73.93 %**	17 0.35 0.73 88.02 % 78.06 %	14 0.49 0.63 83.03 % **79.01 %**	10 0.40 0.62 86.20 % **80.81 %**
L1 0.01	3 3.77 3.79 9.92 % 10.14 %	3 2.83 2.84 9.93 % 10.14 %	3 4.43 4.46 9.94 % 10.14 %	3 3.73 3.75 9.95 % 10.14 %	3 2.55 2.55 9.91 % 10.14 %	3 3.30 3.32 9.93 % 10.14 %
L1 0.001	3 2.45 2.45 9.92 % 10.14 %	3 2.36 2.36 9.88 % 10.14 %	3 2.51 2.52 9.95 % 10.14 %	3 2.44 2.44 9.91 % 10.14 %	3 2.33 2.33 9.91 % 10.14 %	3 2.40 2.40 9.93 % 10.14 %
L1 0.0001	3 2.32 2.32 9.92 % 10.14 %	20 1.06 1.14 67.04 % **64.45 %**	3 2.32 2.32 9.94 % 10.14 %	19 1.12 1.25 66.90 % 63.21 %	3 2.31 2.31 9.92 % 10.14 %	3 2.31 2.31 9.92 % 10.14 %

Table 2: The best epoch, training loss, validation loss, training accuracy and validation accuracy of different models with different regularization techniques with various parameters – part 1

	DN – SP 16-256 10	**DN – MP 16-256 128, 10**	**DN – MP 32-512 256, 10**	**VGG – MP 32-64 256, 128, 10**	**VGG – MP 16-64 128, 10**	**VGG – MP 32-128 256, 10**
L2 0.01	3 2.30 2.30 9.92 % 10.14 %	3 2.30 2.30 9.91 % 10.14 %	3 2.30 2.30 9.89 % 10.14 %	3 2.30 2.30 9.94 % 10.14 %	3 2.30 2.30 9.92 % 10.14 %	3 3.30 3.32 9.93 % 10.14 %
L2 0.001	3 2.30 2.30 9.92 % 10.14 %	18 0.99 1.09 74.11 % 71.20 %	3 2.30 2.30 9.92 % 10.14 %	18 0.86 1.18 81.85 % 72.08 %	16 0.91 1.04 76.81 % **72.66 %**	3 2.40 2.40 9.93 % 10.14 %
L2 0.0001	3 2.30 2.30 9.92 % 10.14 %	13 0.61 1.12 83.20 % 70.08 %	7 0.80 0.94 75.74 % 71.07 %	8 0.54 1.12 86.46 % 71.39 %	12 0.53 1.02 84.99 % **72.65 %**	3 2.31 2.31 9.92 % 10.14 %
L1 0.01 L2 0.01	3 3.79 3.80 9.92 % 10.14 %	3 2.84 2.84 9.92 % 10.14 %	3 4.45 4.47 9.95 % 10.14 %	3 3.74 3.75 9.94 % 10.14 %	3 2.56 2.56 9.92 % 10.14 %	3 3.32 3.32 9.93 % 10.14 %
L1 0.001 L2 0.001	3 2.45 2.45 9.92 % 10.14 %	3 2.36 2.36 9.93 % 10.14 %	3 2.51 2.52 9.94 % 10.14 %	3 2.44 2.44 9.92 % 10.14 %	3 2.33 2.33 9.94 % 10.14 %	3 2.40 2.40 9.92 % 10.14 %
L1 0.0001 L2 0.0001	3 2.32 2.32 9.92 % 10.14 %	16 1.19 1.21 62.12 % 61.87 %	3 2.32 2.32 9.95 % 10.14 %	20 1.00 1.21 73.31 % 67.30 %	15 1.04 1.12 70.50 % **67.79 %**	3 2.31 2.31 9.92 % 10.14 %

Table 3: The best epoch, training loss, validation loss, training accuracy and validation accuracy of different models with different regularization techniques with various parameters – part 2

	type	**kernel size / pooling size / dropout rate**	**number of kernels / number of neurons**	**input shape**	**output shape**	**activation function**	**number of parameters**
1	CONV	3x3	32	[B, 32, 32, 1]	[B, 32, 32, 32]	ReLu	320
2	BN	—	—	[B, 32, 32, 32]	[B, 32, 32, 32]	—	128
3	CONV	3x3	32	[B, 32, 32, 32]	[B, 32, 32, 32]	ReLu	9248
4	BN	—	—	[B, 32, 32, 32]	[B, 32, 32, 32]	—	128
5	MP	2x2	—	[B, 32, 32, 32]	[B, 16, 16, 32]	—	0
6	CONV	3x3	64	[B, 16, 16, 32]	[B, 16, 16, 64]	ReLu	18496
7	BN	—	—	[B, 16, 16, 64]	[B, 16, 16, 64]	—	256
8	CONV	3x3	64	[B, 16, 16, 64]	[B, 16, 16, 64]	ReLu	36928
9	BN	—	—	[B, 16, 16, 64]	[B, 16, 16, 64]	—	256
10	MP	2x2	—	[B, 16, 16, 64]	[B, 8, 8, 64]	—	0
11	CONV	3x3	128	[B, 8, 8, 64]	[B, 8, 8, 128]	ReLu	73856
12	BN	—	—	[B, 8, 8, 128]	[B, 8, 8, 128]	—	512
13	CONV	3x3	128	[B, 8, 8, 128]	[B, 8, 8, 128]	ReLu	147584
14	BN	—	—	[B, 8, 8, 128]	[B, 8, 8, 128]	—	512
15	MP	2x2	—	[B, 8, 8, 128]	[B, 4, 4, 128]	—	0
16	FT	—	—	[B, 4, 4, 128]	[B, 2048]	—	0
17	DO	0.4	—	[B, 2048]	[B, 2048]	—	0
18	DENS	—	256	[B, 2048]	[B, 256]	ReLu	524544
19	DO	0.4	—	[B, 256]	[B, 256]	—	0
20	DENS	—	10	[B, 256]	[B, 10]	softmax	2570
						total number of parameters	**815 338**
						total number of trainable parameters	**814 442**
						total number of non-trainable parameters	**896**

Table 4: Per layer architecture summary of the final model

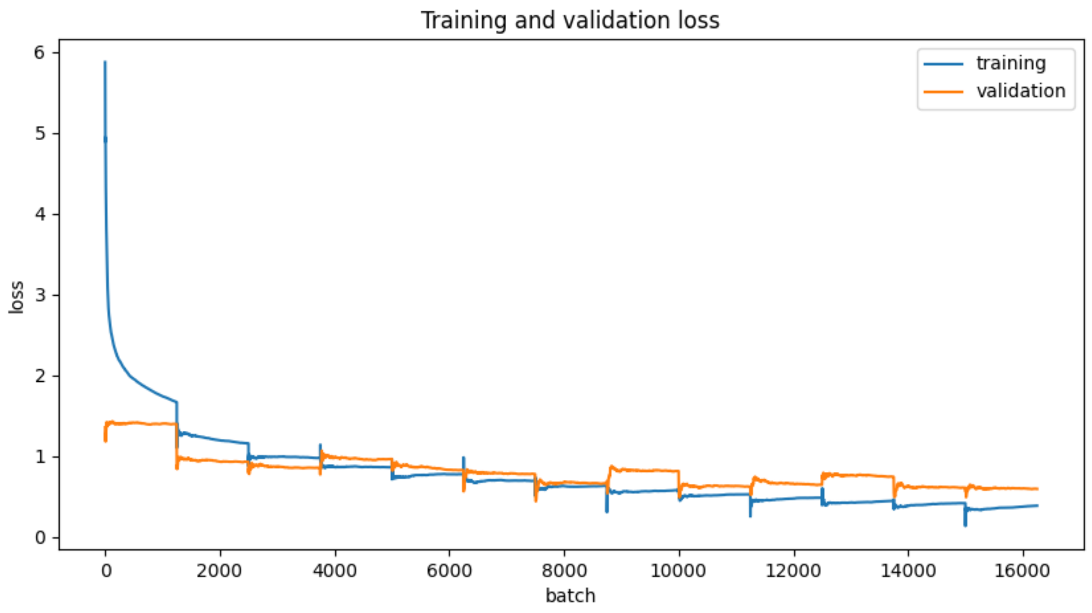
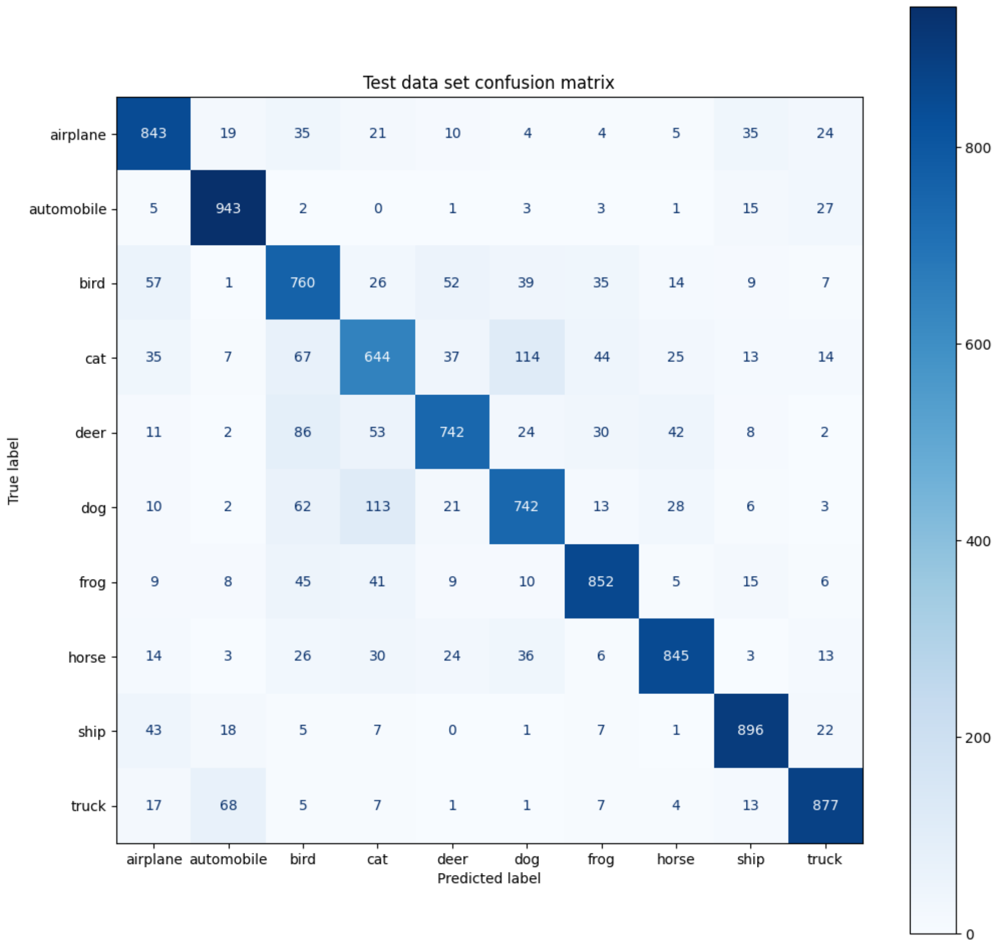


Figure 1: Final model development of training and validation loss during training



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	843	19	35	21	10	4	4	5	35	24
automobile	5	943	2	0	1	3	3	1	15	27
bird	57	1	760	26	52	39	35	14	9	7
cat	35	7	67	644	37	114	44	25	13	14
deer	11	2	86	53	742	24	30	42	8	2
dog	10	2	62	113	21	742	13	28	6	3
frog	9	8	45	41	9	10	852	5	15	6
horse	14	3	26	30	24	36	6	845	3	13
ship	43	18	5	7	0	1	7	1	896	22
truck	17	68	5	7	1	1	7	4	13	877

Figure 2: Confusion matrix of the final model on the test data set

	training data set accuracy	**test data set accuracy**	**perturbed data set accuracy**
unchanged final model	87.82 %	81.44 %	63.58 %
max pooling layers replaced by average pooling layers	89.71 %	81.41 %	62.19 %
l_2-regularization in conv. layers, dropout replaced by l_2-regularization, batch norm. after hidden fully conn. layer	93.32 %	79.67 %	62.14 %
l_2-regularization in conv. layers, 0.25 dropout and l_2-regularization, batch norm. after hidden fully conn. layer	89.41 %	78.86 %	56.50 %
unchanged final model trained on a perturbed training data set by salt and pepper noise and gaussian noise	90.52 %	79.86 %	**78.24 %**
unchanged final model trained on a perturbed training data set by rotation and color distortions	63.56 %	**82.91 %**	71.68 %

Table 5: Accuracies of the final model with modifications and after training on perturbed training data sets on the training, test and perturbed test data sets.

References

[1] Redmon, J. and Farhadi, A. YOLOv3: An Incremental Improvement. *CoRR*. 2018, abs/1804.02767. Available at: http://arxiv.org/abs/1804.02767.

[2] Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*. 2015, abs/1409.1556. Available at: http://arxiv.org/abs/1409.1556.