NN for CIFAR-10/CIFAR-100

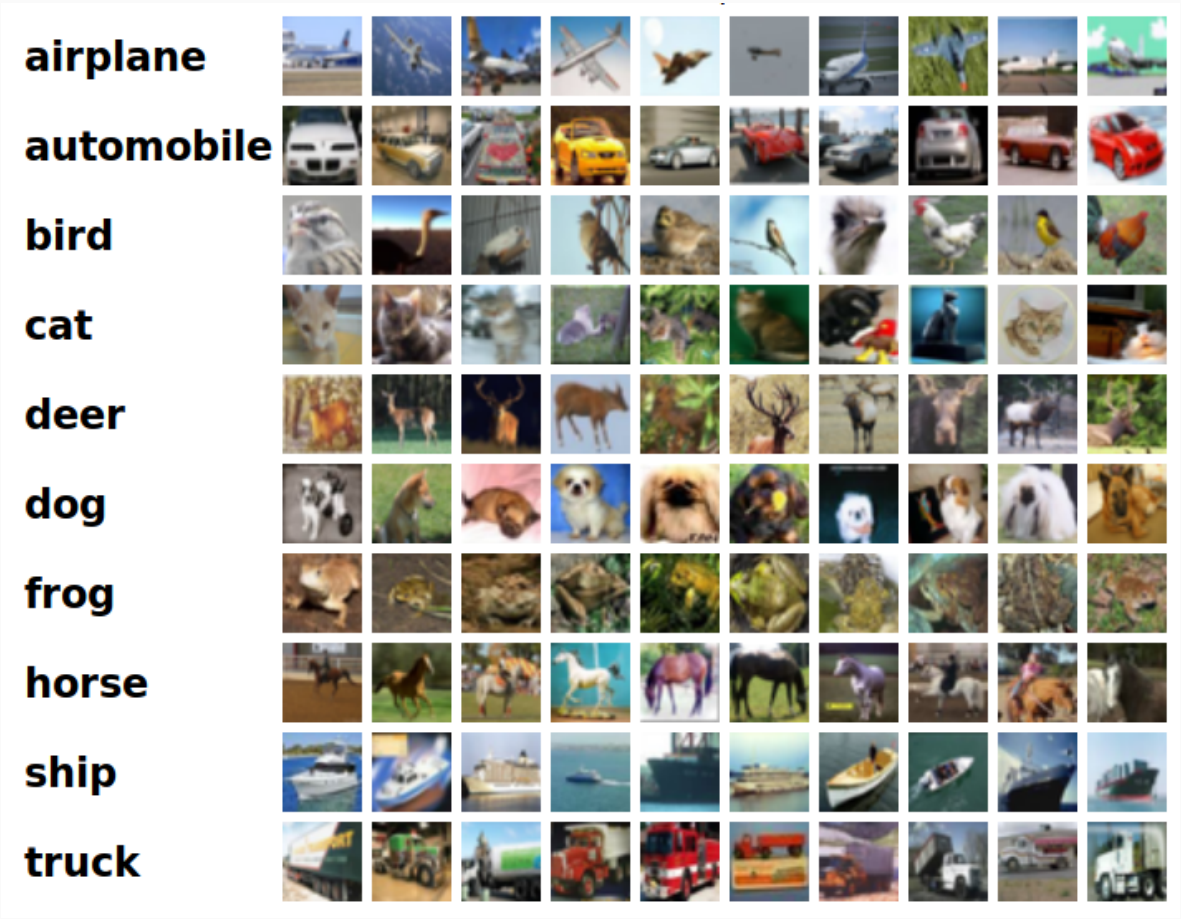
Matthias Ollech, Jannik Michels

July 18, 2019

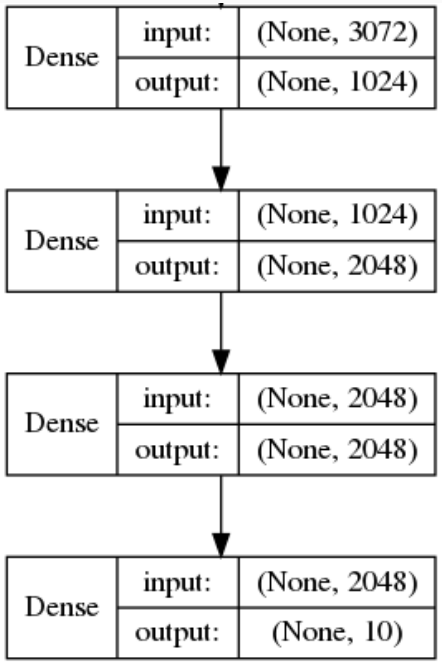
Dataset CIFAR-10

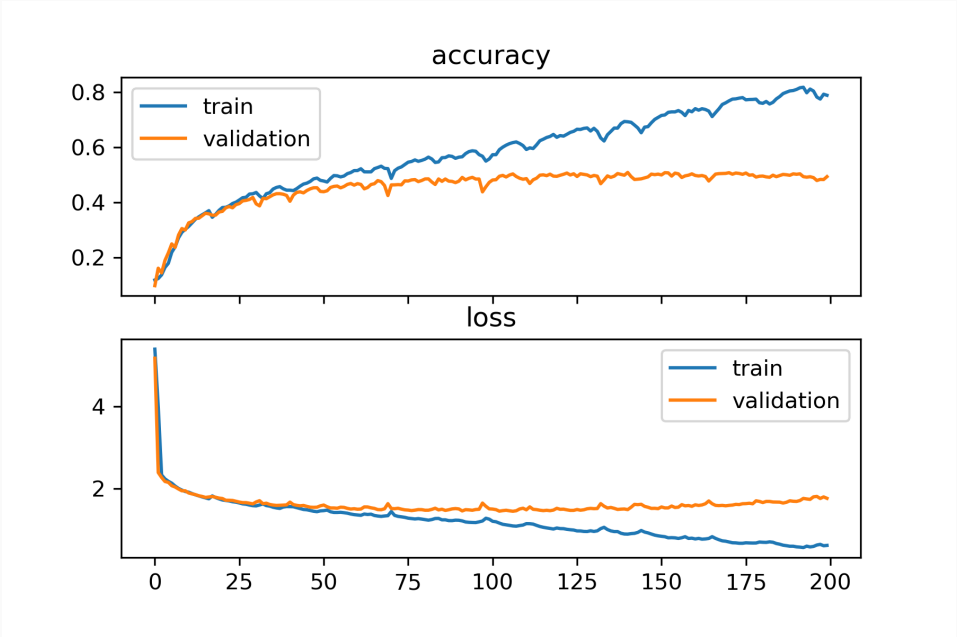
CIFAR-10

Consists of 50000 32x32 images in 10 classes



Simple NN

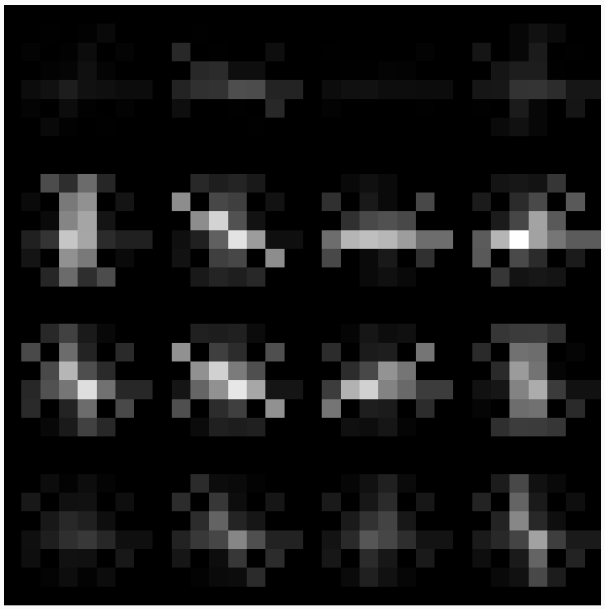




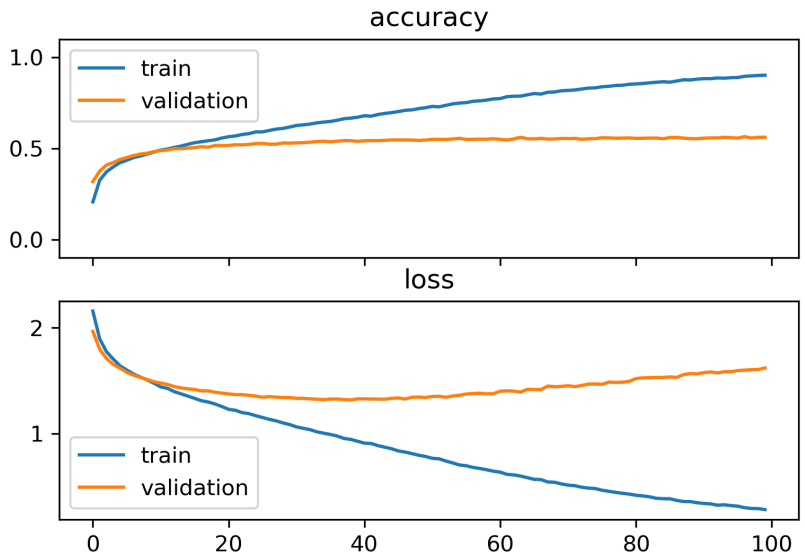
Accuracy after 100 epochs:

- Train: 90.28%
- Validation: 56.17%

HoG



Same NN with dropout of 0.3 after each layer.



Accuracy after 100 epochs:

- Train: 70.48%
- Validation: 62.70%

CNN

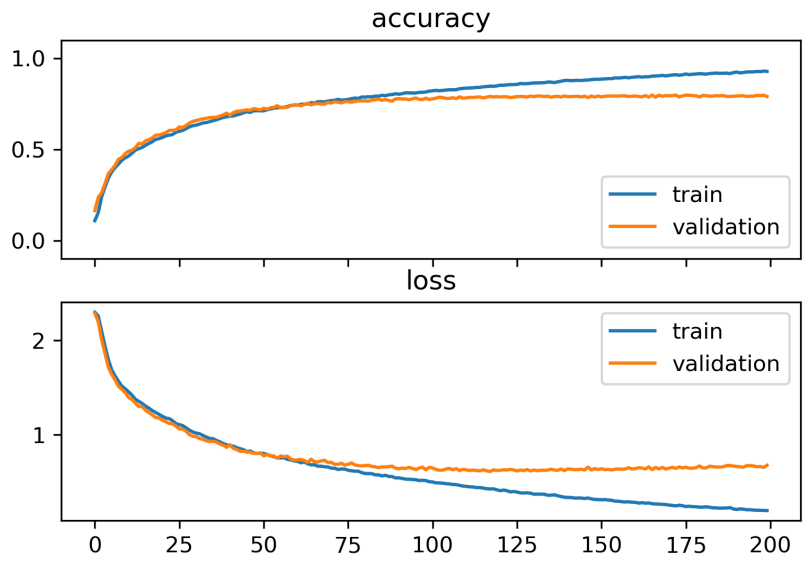
Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 32)	1568
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 32)	0
dropout_7 (Dropout)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	32832
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 64)	0
dropout_8 (Dropout)	(None, 8, 8, 64)	0
conv2d_3 (Conv2D)	(None, 8, 8, 64)	65600
max_pooling2d_3 (MaxPooling2	(None, 4, 4, 64)	0
flatten_1 (Flatten)	(None, 1024)	0
dropout_9 (Dropout)	(None, 1024)	0
dense_13 (Dense)	(None, 2048)	2099200
dropout_10 (Dropout)	(None, 2048)	0
dense_14 (Dense)	(None, 2048)	4196352
dropout_11 (Dropout)	(None, 2048)	0
dense_15 (Dense)	(None, 10)	20490

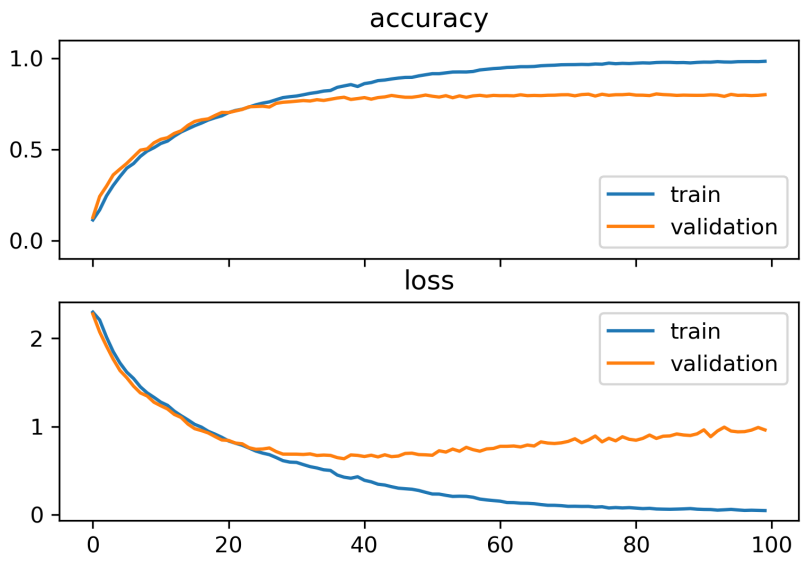
Total params: 6,416,042
Trainable params: 6,416,042
Non-trainable params: 0

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 32, 32, 96)	7296
max_pooling2d_4 (MaxPooling2	(None, 15, 15, 96)	0
dropout_12 (Dropout)	(None, 15, 15, 96)	0
conv2d_5 (Conv2D)	(None, 15, 15, 128)	307328
max_pooling2d_5 (MaxPooling2	(None, 7, 7, 128)	0
dropout_13 (Dropout)	(None, 7, 7, 128)	0
conv2d_6 (Conv2D)	(None, 7, 7, 256)	819456
max_pooling2d_6 (MaxPooling2	(None, 3, 3, 256)	0
dropout_14 (Dropout)	(None, 3, 3, 256)	0
flatten_2 (Flatten)	(None, 2304)	0
dense_16 (Dense)	(None, 2048)	4720640
dropout_15 (Dropout)	(None, 2048)	0
dense_17 (Dense)	(None, 2048)	4196352
dropout_16 (Dropout)	(None, 2048)	0
dense_18 (Dense)	(None, 10)	20490

Total params: 10,071,562
Trainable params: 10,071,562
Non-trainable params: 0

CNN - Test





Accuracy after 100 epochs:

- Train: 82.35%
- Validation: 78.24%

Accuracy after 100 epochs:

- Train: 98.53%
- Validation: 80.30%

ZCA I

Let X be a vector, $\varepsilon > 0$ a real number and $USV = Cov(X)$ be the singular value decomposition of the covariance matrix, then

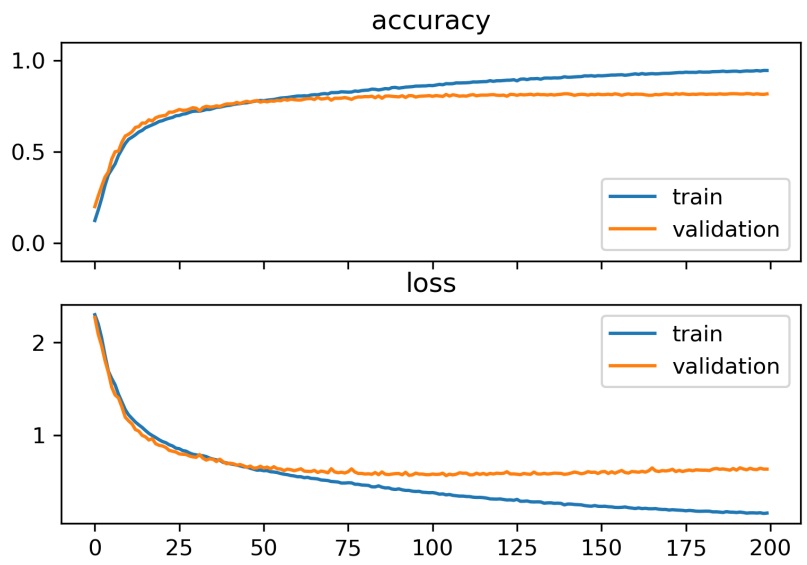
$$X_{ZCA(\varepsilon)} := X \cdot U \cdot \sqrt{S + \mathbf{1}\epsilon}^{-1} \cdot U^T$$

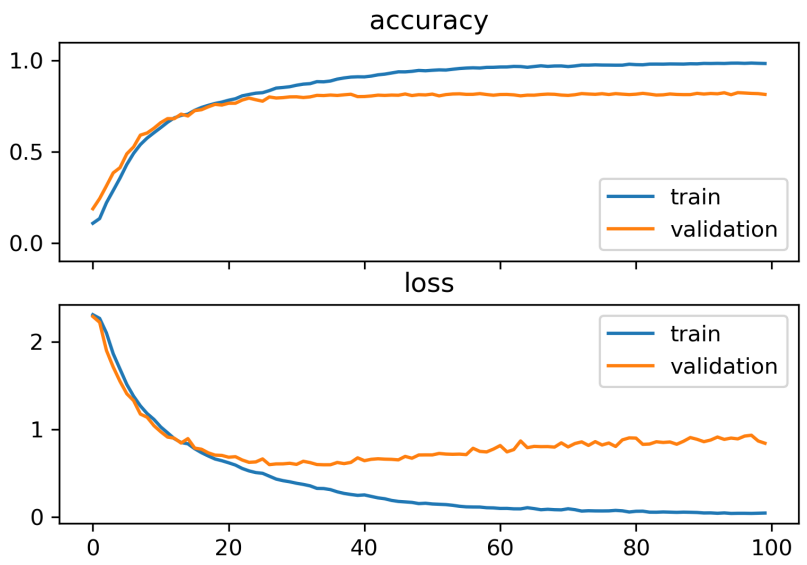
Original $\varepsilon = 10^{-1}$ $\varepsilon = 10^{-2}$ $\varepsilon = 10^{-3}$ $\varepsilon = 10^{-4}$

Original $\varepsilon = 10^{-1}$ $\varepsilon = 10^{-2}$ $\varepsilon = 10^{-3}$ $\varepsilon = 10^{-4}$

ZCA - Test

It turns out, that $\varepsilon = 0.01$ is the best choice.





Accuracy after 200 epochs:

- Train: 94.70%
- Validation: 81.31%

Accuracy after 100 epochs:

- Train: 98.89%
- Validation: 82.09%

Testing the best

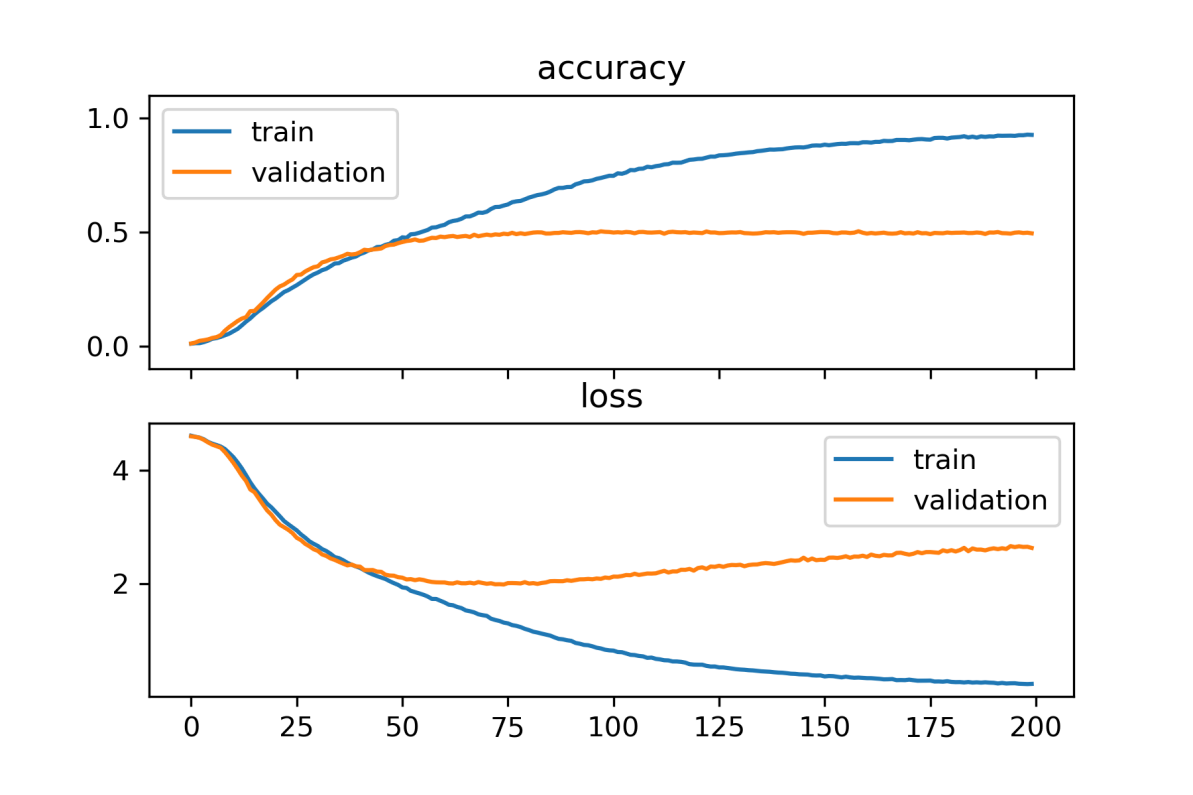
Here we applied the CNN with the best results to the test data:

Label/Prediction	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	823	11	41	20	16	11	7	11	49	25
Automobile	5	934	2	3	1	5	7	4	22	31
Bird	28	3	734	32	42	52	38	13	8	2
Cat	13	4	42	653	35	179	53	14	8	15
Deer	11	1	48	43	803	28	35	22	4	2
Dog	4	1	22	119	32	776	20	45	4	2
Frog	2	3	26	28	8	19	891	0	3	0
Horse	6	0	27	26	35	59	8	801	32	12
Ship	28	17	6	8	3	3	9	1	915	13
Truck	15	67	4	13	3	4	7	9	16	884

We got an accuracy of 82,14%.

Dataset CIFAR-100

CIFAR-100 test



Accuracy after 100 epochs:

- Train: 84.39%
- Validation: 52.49%
- Test: 52.29%

Technical aspects

Technical aspects

- optimizer: adam (faster) vs. sgd (higher accuracy)
- batch size: 10000 (less oszilation/higher accuracy) vs. 50 (faster)
- computing device: CPU $\left(180 \frac{s}{\text{Epoch}}\right)$ vs. GPU $\left(6 \frac{s}{\text{Epoch}}\right)$

Questions?