UNIVERSITÄT BERN

CAS Applied Data Science

Module 6 –Convolutional Neural Networks Application

Enhanced image classification in the CIFAR10 dataset

Frederic Bärtl, Nicolas Gaillard

13th March 2023, Bern

Structure of the presentation

1. What is the data about ?
2. Looking back at our work from M3
3. How to enhance the results ?
4. Application of CNN
5. Enhanced CNN model with data augmentatiion

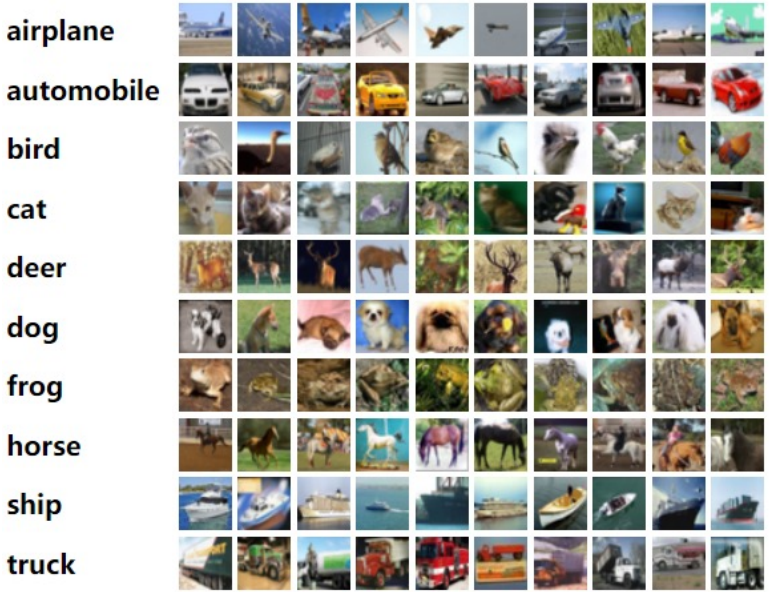
Structure of the presentation

1. What is the data about ?
2. Looking back at our work from M3
3. How to enhance the results ?
4. Enhanced CNN model with data augmentation
5. Summary

1. Data Analysis (from M3)

Key points about the used data:

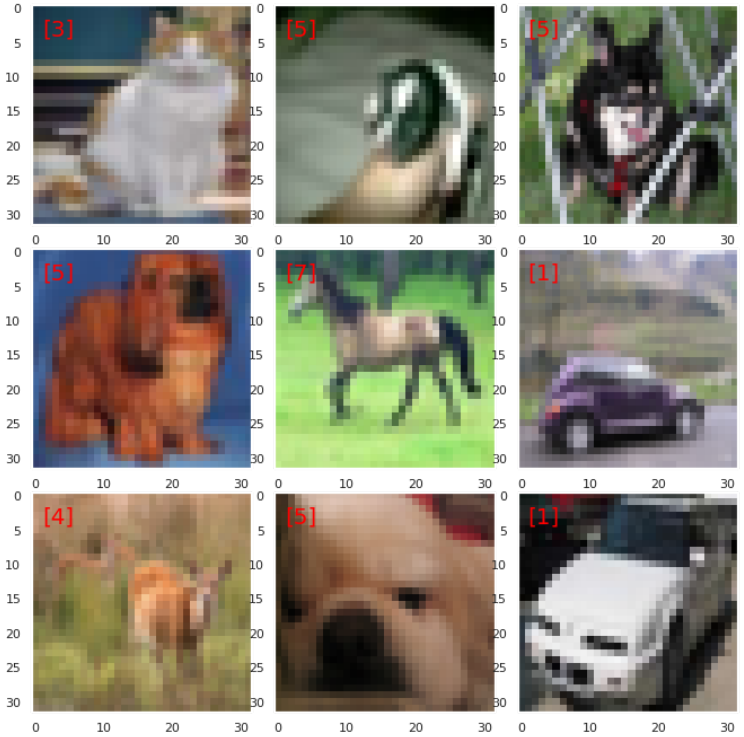
- Dataset consists of 60000 images of 32x32 pixels
- In all models, we split this into a training set of 50000 and a test set of 10000 images
- Dataset is labeled and has 10 different categories (see on the right)



1. Data Analysis (from M3)

Key points about the used data:

- Due to the low resolution of the images, humans can derive the correct category in 94% of the cases
- Thus to make a model that is better than the human eye, accuracy must be > 94 %



UNIVERSITÄT BERN

Structure of the presentation

1. What is the data about ?
2. Looking back at our work from M3
3. How to enhance the results ?
4. Enhanced CNN model with data augmentation
5. Summary

13. März 2023, Bern

Lookback at M3: Random forest

UNIVERSITÄT BERN

Random forest decision tree classifier model

N_est Setting	Max_depth setting	Accuracy Training set	Accuracy Test set
10	5	33%	32%
100	100	42%	39%
200	700	59%	43%
500	5000	74%	**45%**

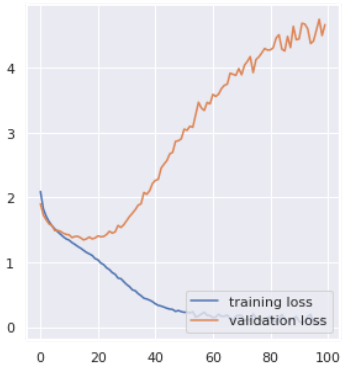
The highest accuracy we achieved using «Random forest» methods was approx. 45 %

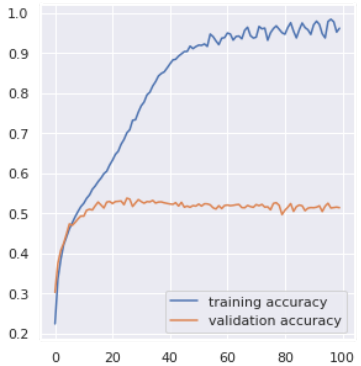
Lookback at M3: Neural Network application

We re-did the M3 analysis after the feedback session and tried to optimize our accuracy result:

```
model = tf.keras.models.Sequential([
tf.keras.layers.Flatten(input_shape=(32, 32, 3)),
tf.keras.layers.Dense(1500, activation='relu'),
tf.keras.layers.Dense(1000, activation='relu'),
tf.keras.layers.Dense(400, activation='relu'),
tf.keras.layers.Dense(50, activation='relu'),
#tf.keras.layers.Dense(50, activation='relu'),
tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
loss='sp
arse_categorical_crossentropy',
metrics=['accuracy'])
```





The highest accuracy we achieved with a “classical” neural network model is approx. 50 % accuracy. But tendency of overfitting not efficient on independent (validation) data

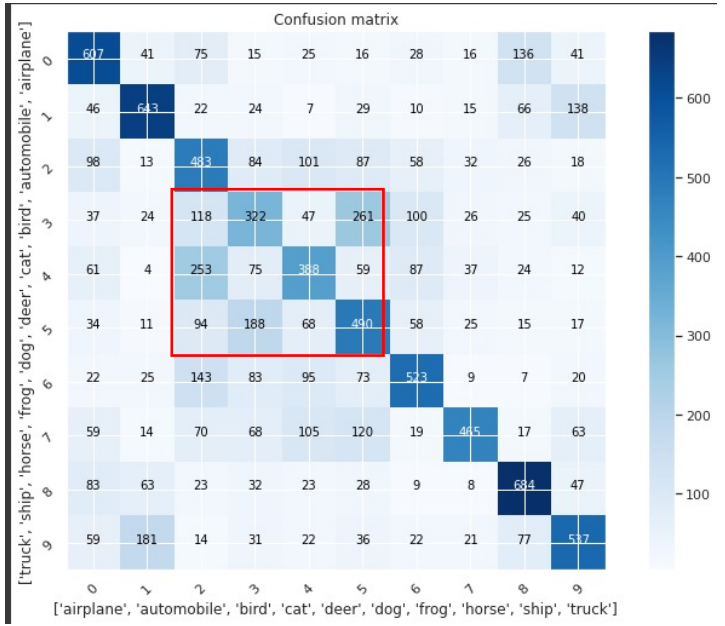
Lookback at M3: Isolated problem

UNIVERSITÄT BERN

What is the model getting wrong ?

> The model is unsure about very specific categories

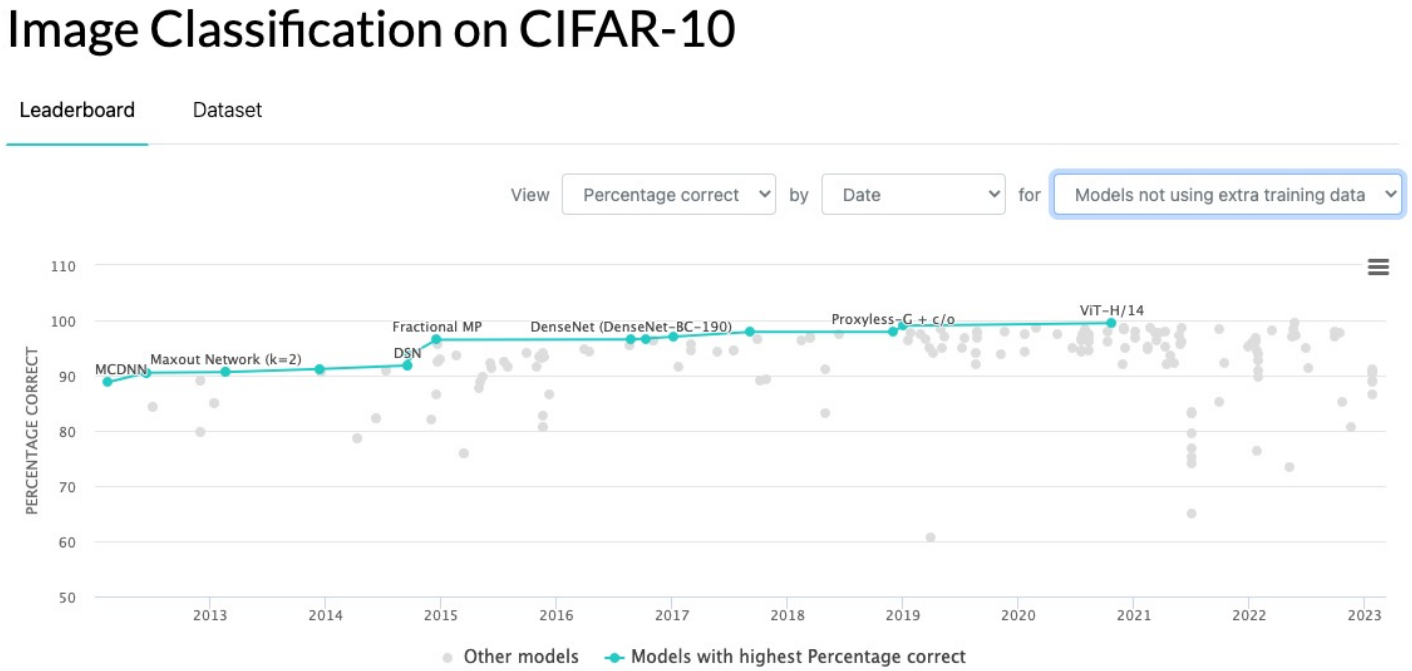
> The main problem is around dog, cats and deer



Structure of the presentation

1. What is the data about ?
2. Looking back at our work from M3
3. How to enhance the results ?
4. Enhanced CNN model with data augmentation
5. Summary

What performance is possible ?



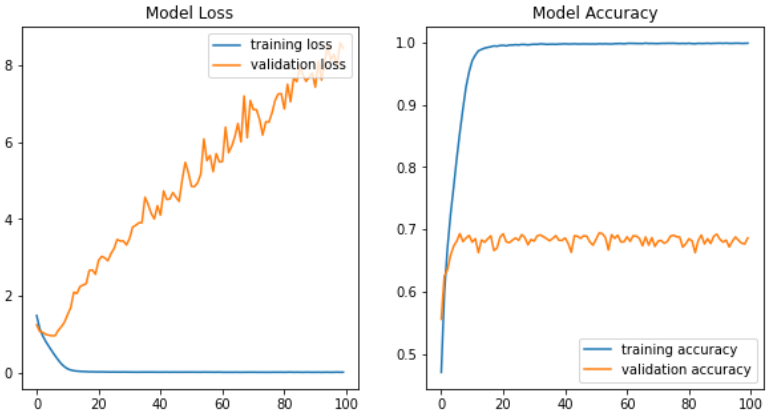
Best CNN reaches approx. 93 % accuracy (ranked 98th) → Our new benchmark

1. CNN application – select learning rate

We changed the learning rate to see if this has a positive impact:

Learning rate = 0.001

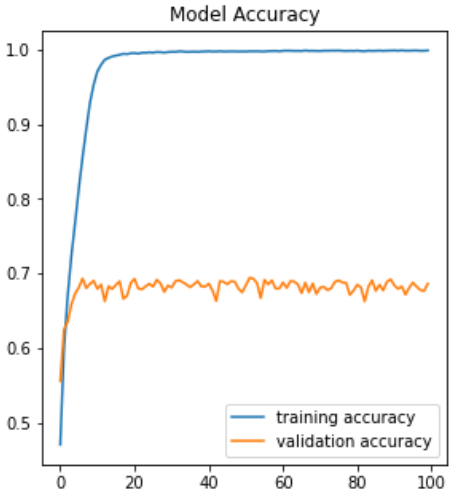
Learning rate = 0.0001



1. CNN application – First try

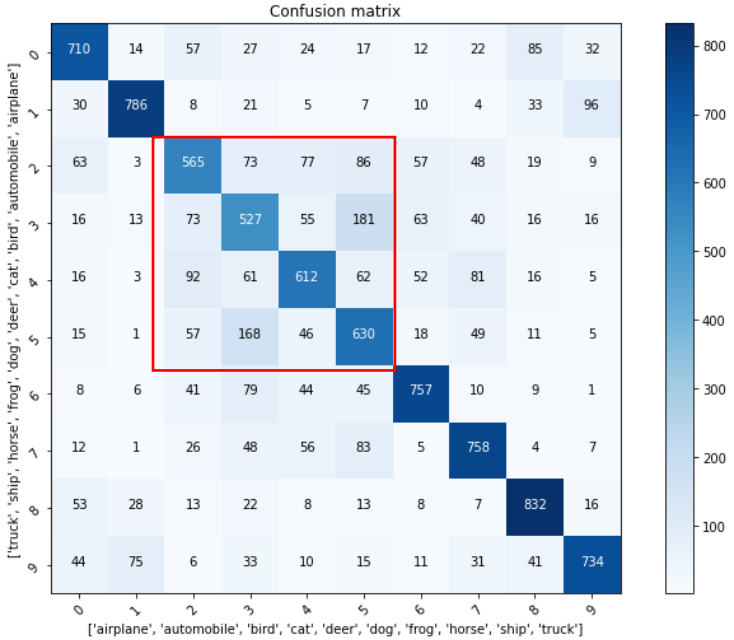
First try: Model with 4 Convolution layers with 32 to 64 neurons

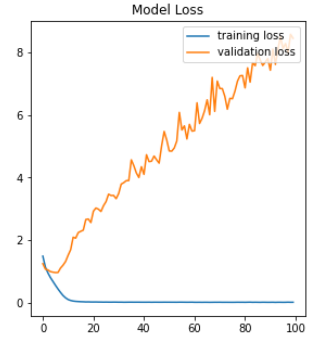
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
conv2d_1 (Conv2D)	(None, 30, 30, 32)	9248
conv2d_2 (Conv2D)	(None, 30, 30, 64)	18496
conv2d_3 (Conv2D)	(None, 28, 28, 64)	36928
flatten (Flatten)	(None, 50176)	0
dense (Dense)	(None, 512)	25690624
dense_1 (Dense)	(None, 10)	5130
activation (Activation)	(None, 10)	0

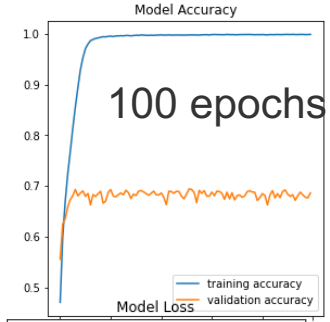


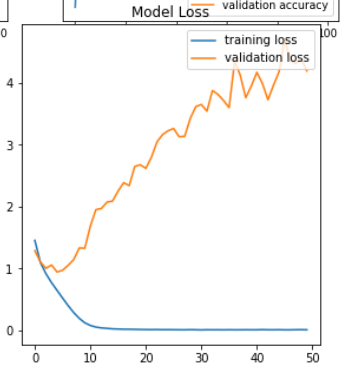
1. CNN application – Ideal epochs

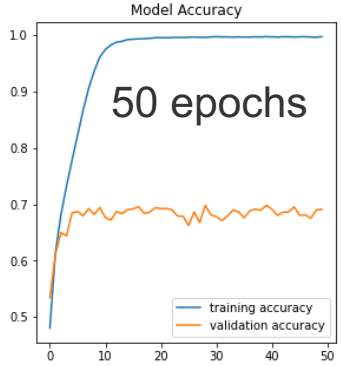
First try: Model with 4 Convolution layers with 32 to 64 neurons











Structure of the presentation

1. What is the data about ?
2. Looking back at our work from M3
3. How to enhance the results ?
4. Enhanced CNN model with data augmentation
5. Summary

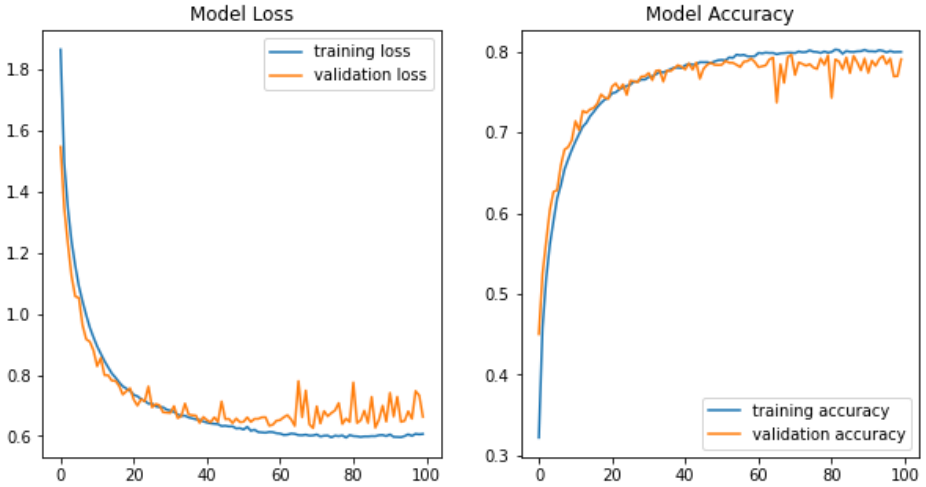
2. Enhanced CNN application

We added:

- **Max.pooling** to reduce the output dimensions per layer
- **dropout** to better train the model by randomly blanking out certain paths (0,0.25,0,0.25,05)

→ Results slightly improved (see performance matrix for comparison)

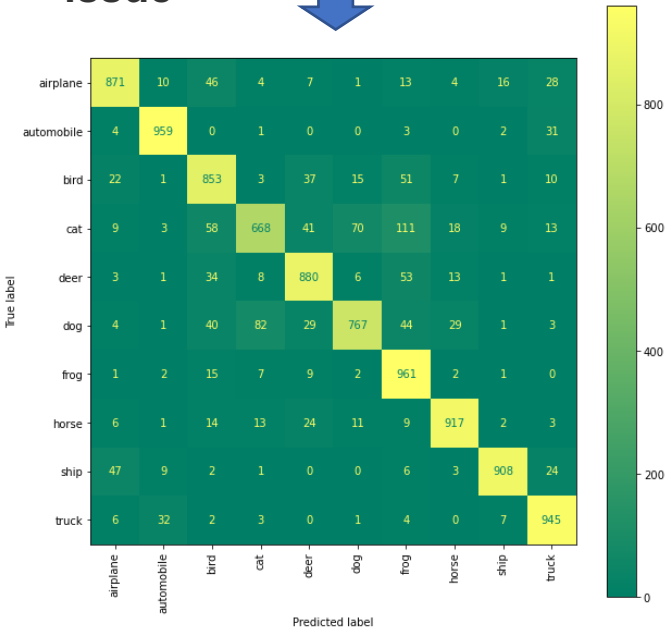
→ Conversion after 40-50 epochs thus further adjustments only done with 50 epochs for efficiency reasons

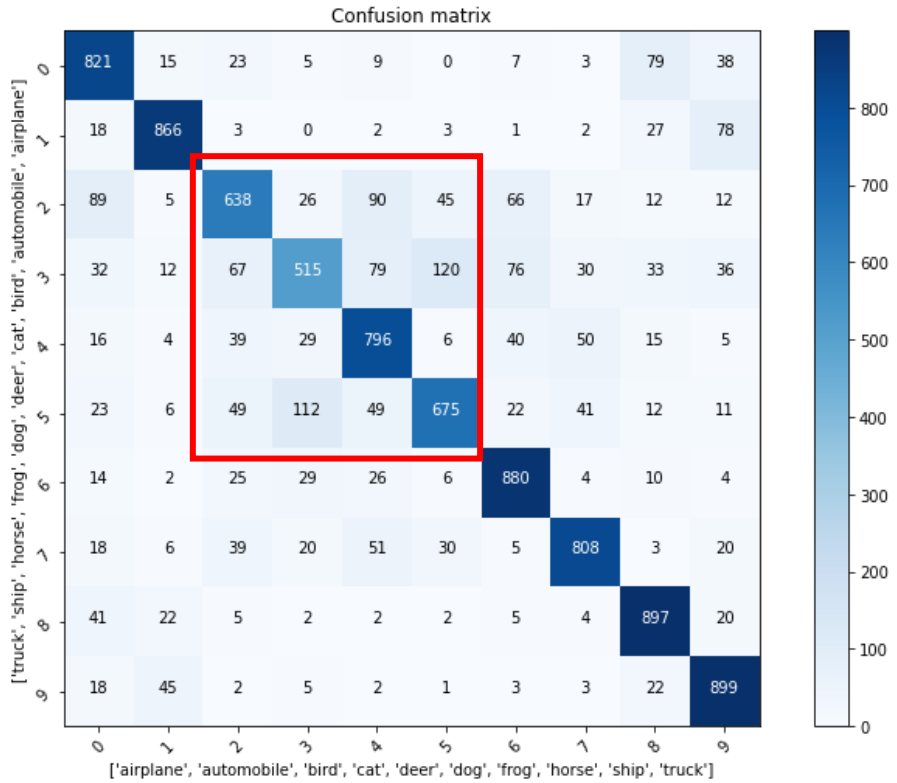


Accuracy increases by 10% points to approx. 78 %

2. Confusion Matrix --> mixed results

- **Only 4 categories below 80% precision**
- **Even best models face the same issue**

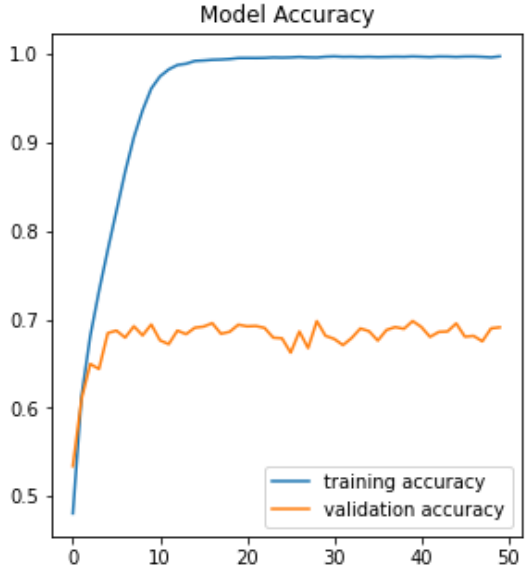




2. Enhanced CNN application

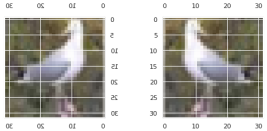
We added:

- **Strides = 2 to decrease the amount of parameters leads to a decrease in accuracy to 69 %**



2. Enhanced CNN application

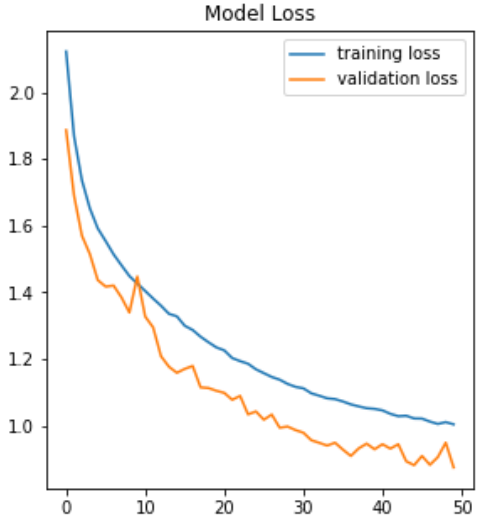
We added:

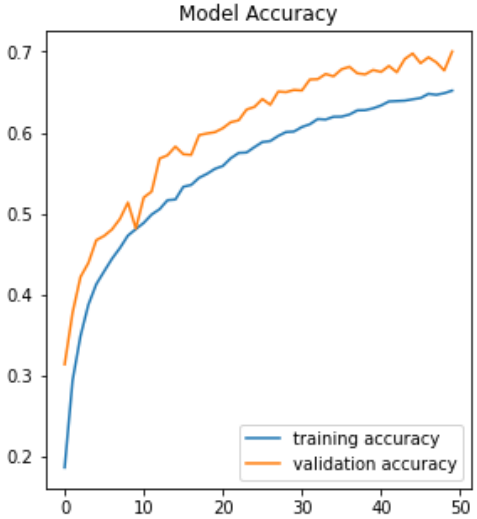


- **Height_shift_range = 0.1**
- **Horizontal_flip = True**

Accuracy is reduced from 79 % to 66 % thus manipulation of the images does not enhance performance of the model.

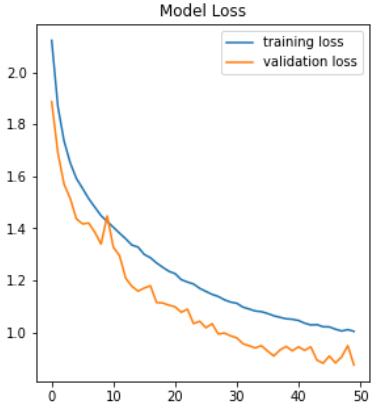
Public discussion indicates drop out rate changes might be a better vehicle

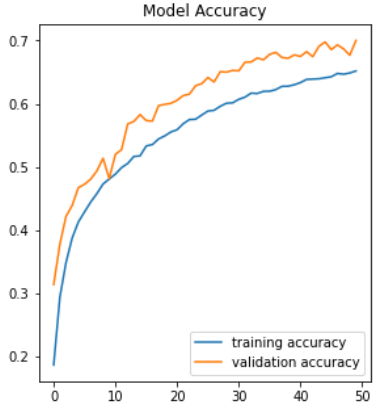


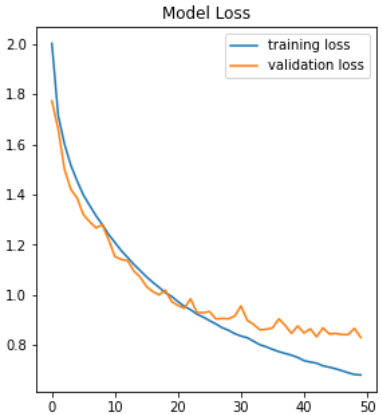


2. Enhanced CNN – Adjusted drop out rates

- 0
- 0.35
- 0
- 0.35
- 0.5







- 0
- 0.1
- 0
- 0.1
- 0.4

Best result

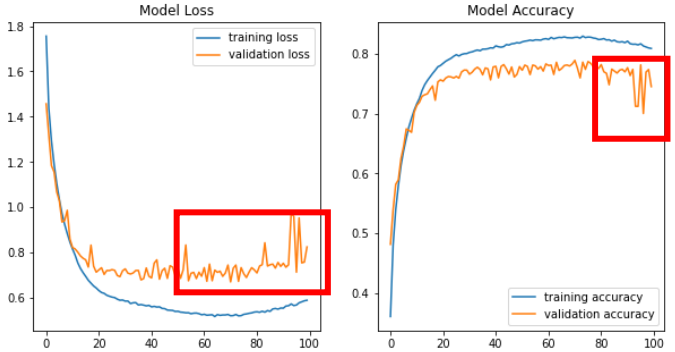
→ Lets build the combination of our “ideal” adjustments

3. Enhanced CNN – Ideal combination

- **Learning rate 0.0001**
- **Max pooling (2,2)**
- **No strides**
- **No image adjustments**
- **Low dropout rates**

50 epochs → 77 % accuracy

100 epochs → 78.5 % accuracy

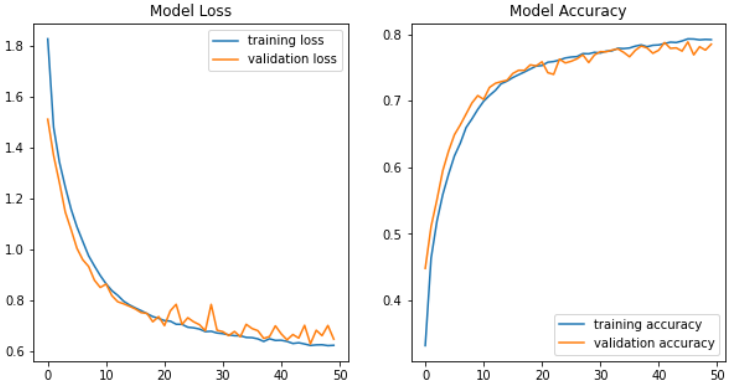


Does not look “ideal” to us ….

3b. CNN – Ideal drop out rates

- **Learning rate 0.0001**
- **Max pooling (2,2)**
- **No strides**
- **No image adjustments**
- **Dropout: 0,0.2,0,0.2,05**

50 epochs → 78.5 % accuracy



100 epochs → 79.0 % accuracy

How does the precision looks like in comparison ?

3c. CNN – Adjusted convolution filter

Increasing convolution filter from (3,3) to (5,5) incorporating more information of neighboring pixels.

50 epochs → 79.0 % accuracy

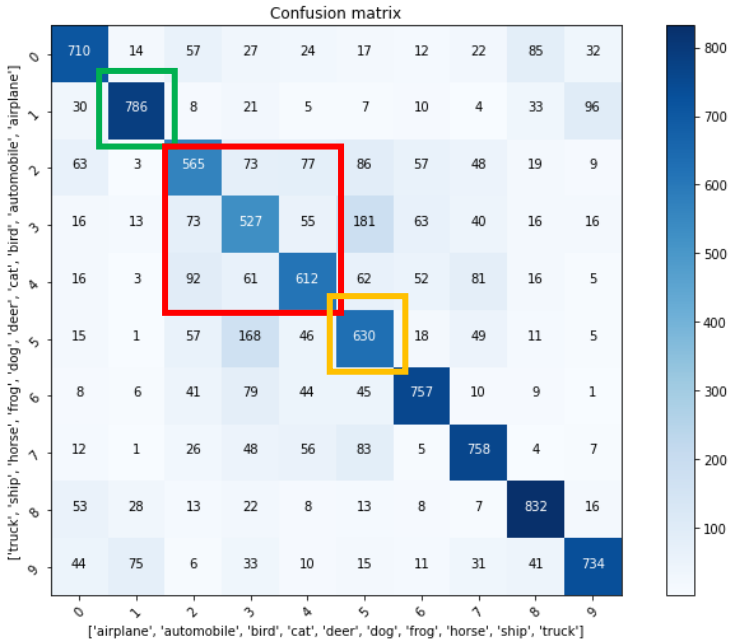
100 epochs → 80.0 % accuracy

Training accuracy increases but validation accuracy stays the same. We assume this is because of the small pixel size our out images and bigger pictures might benefit from increased filters.

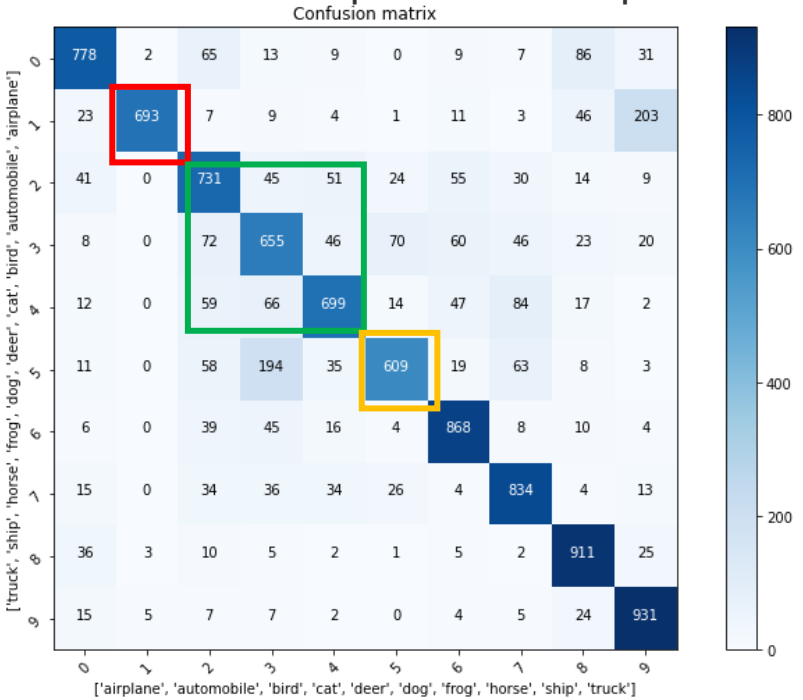
3d. CNN – Ideal combination, better but...

UNIVERSITÄT BERN

Initial model (no dropout) - 69%



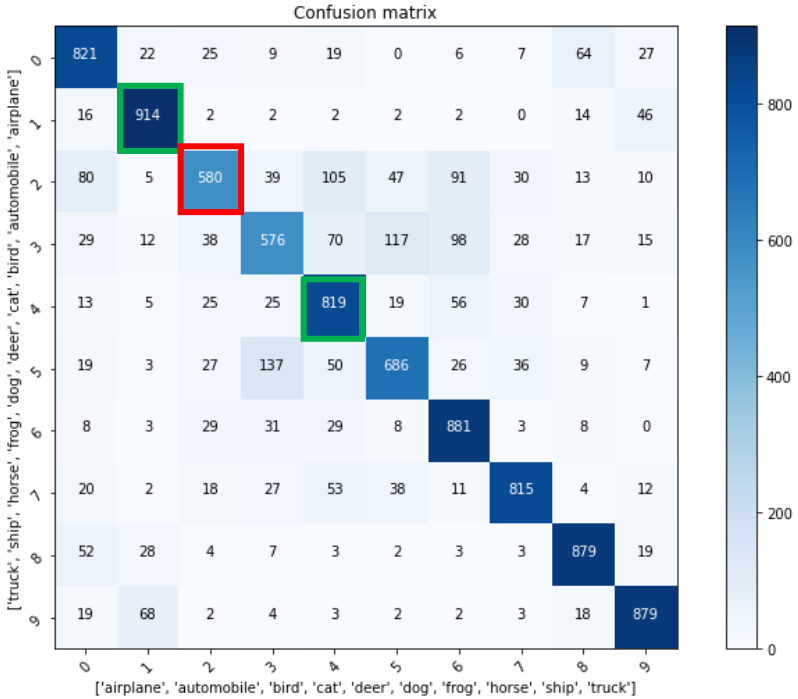
3b model with 100 epochs and dropout – 79 %



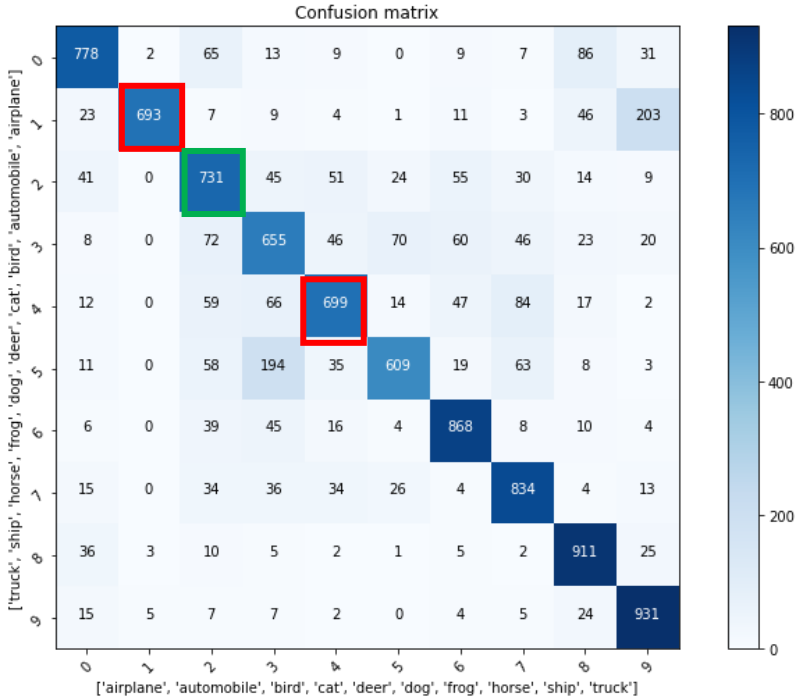
3d. Looks like random effects are strong..

UNIVERSITÄT BERN

3b model with 50 epochs



3b model with 100 epochs



Marked = more than 10 percentage point precision difference

Structure of the presentation

1. What is the data about ?
2. Looking back at our work from M3
3. How to enhance the results ?
4. Enhanced CNN model with data augmentation
5. Summary

Model performance comparison

Test Set	Performance Matrix (accurancy)	100 epochs Learning rate 0.0001	50 epochs Learning rate 0.0001	50 epochs Learning rate 0.001
1	3 Layer model	69 %	Not tested	25 %
2	+Max. Pooling (2,2)& Dropout 0.25	78 %	77.5 %	Not tested
2a	+Stride = 2	69 %	69 %	Not tested
2a+b	+height shift & hori. flip	66 %	66 %	Not tested
2a+c	+ Higher dropout	64 %	64 %	Not tested
2a+d	+ Lower dropout	71 %	71 %	Not tested
3	Back to 3 layer model with low drop out (2+ 2d)	78.5 %	77 %	Not tested
3b	Back to 3 layer model with medium drop out (2+ new setup)	**79.0 %**	78.5 %	Not tested
3c	Increasing the Convolution layer to 5x5	**80.0 %**	**79.0 %**	Not tested

Summary of our ML application

UNIVERSITÄT BERN

Minimal goal:

Our CNN models outperformed our M3 approach substantially (50% vs. 80 %)

What had the biggest impact ?

- Data augmentation failed to improve accuracy
  - Maybe the wrong adjustments ?
  - Would be interesting to see what happens when we only augment the pictures from the low performing categories but we did not know how to do that
- Strides might save computational time but weakens our model by 10 percentage points
- Drop out rate has a massive impact indicating that model is stronger when it needs to “try different paths” to get to a result.
- Filter size might have a substantial impact on images with larger dimension size

→ We will use these techniques for our final classification problem which will be document “pre-classification” of documents gathered during “house searches”

Layer (type)	Output Shape	Param #
conv2d_94 (Conv2D)	(None, 32, 32, 32)	2432
conv2d_95 (Conv2D)	(None, 28, 28, 32)	25632
max_pooling2d_47 (MaxPooling2D)	(None, 14, 14, 32)	0
dropout_47 (Dropout)	(None, 14, 14, 32)	0
conv2d_96 (Conv2D)	(None, 14, 14, 64)	51264
conv2d_97 (Conv2D)	(None, 10, 10, 64)	102464
max_pooling2d_48 (MaxPooling2D)	(None, 5, 5, 64)	0
dropout_48 (Dropout)	(None, 5, 5, 64)	0
flatten_20 (Flatten)	(None, 1600)	0
dense_39 (Dense)	(None, 512)	819712
dropout_49 (Dropout)	(None, 512)	0
dense_40 (Dense)	(None, 10)	5130
activation_15 (Activation)	(None, 10)	0