

# Ausarbeitung Versuch 3 ILS Jan Holderied und Martin Goien

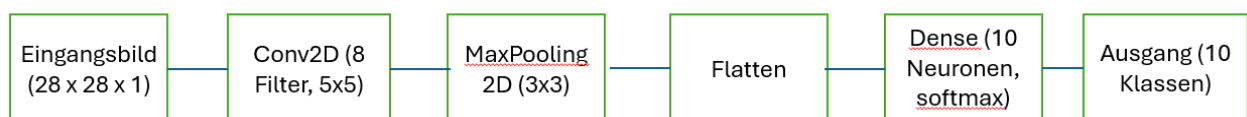
## Aufgabe 1

a)

- **Normalize Image:** Die Pixelwerte der Bilder werden Normalisiert um eine bessere Konvergenz beim Lernen zu bekommen und die Stabilität des Trainings zu erhöhen. Dies wird erreicht in dem man jeden Pixelwert durch 255 dividiert und dann 0,5 subtrahiert, damit liegen alle Pixelwerte zwischen -0,5 und 0,5.
- **Reshape the images:** Es wird ein weitere Dimension/Kanal zu den Daten hinzugefügt, da die CNN Netzwerke das Format (Batch-Größe, Höhe, Breite, Kanäle) erwarten. Es wird einfach ein Kanal mit den Werten 1 hinzugefügt.

b)

- **Hyperparameter**
  - num\_filters = 4: Anzahl der Filter
  - filter\_size = 3: Größe der Filter Layers
  - pool\_size = 2: Größe der Pooling Layers
  - eta = 1e-4: Lernrate für den Gradientenabstieg
  - opt\_alg=SGD(learning\_rate=eta):
  - batchsize=50: n Bilder die pro Gewichtsupdate zum Lernen verwendet werden sollen. 50 pro Batch in diesem Fall.
  - epochs=5: Anzahl der Trainingsepochen.
- **Skizze des Netzwerks**



- **Größe des Netzwerks:**

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
layers_conv (Conv2D)	(None, 24, 24, 8)	208
layers_maxpool (MaxPooling2D)	(None, 8, 8, 8)	0
layers_flatten (Flatten)	(None, 512)	0
layers_dense (Dense)	(None, 10)	5130

- In der Layer Conv2D-Schicht entspricht die Anzahl Physischer-Synapsen nicht der Anzahl unabhängiger trainierbarer Parameter.  
Conv2D-Schicht: Anzahl der Physischen-Synapsen: 200  
Anzahl der unabhängigen trainierbaren Parametern: 208

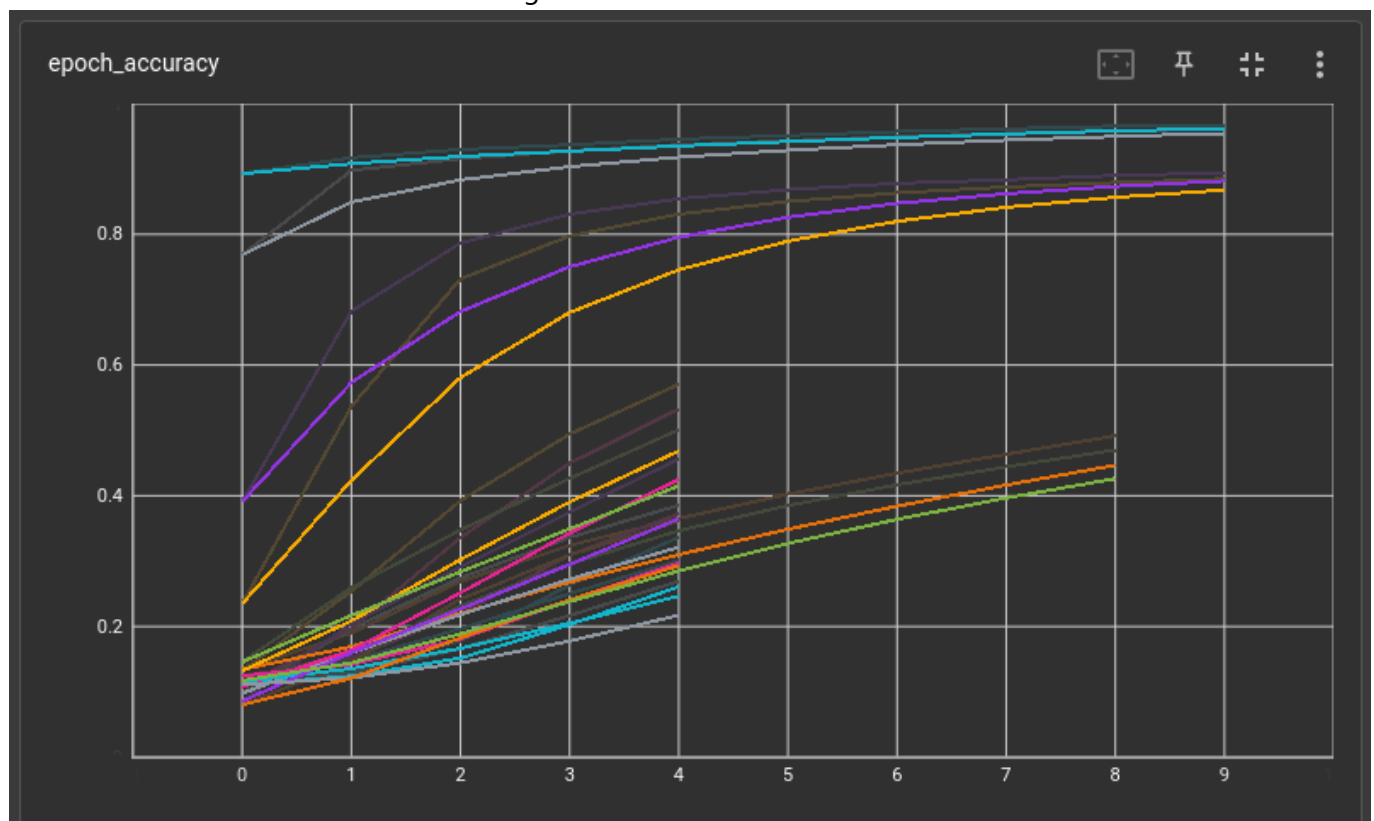
c)

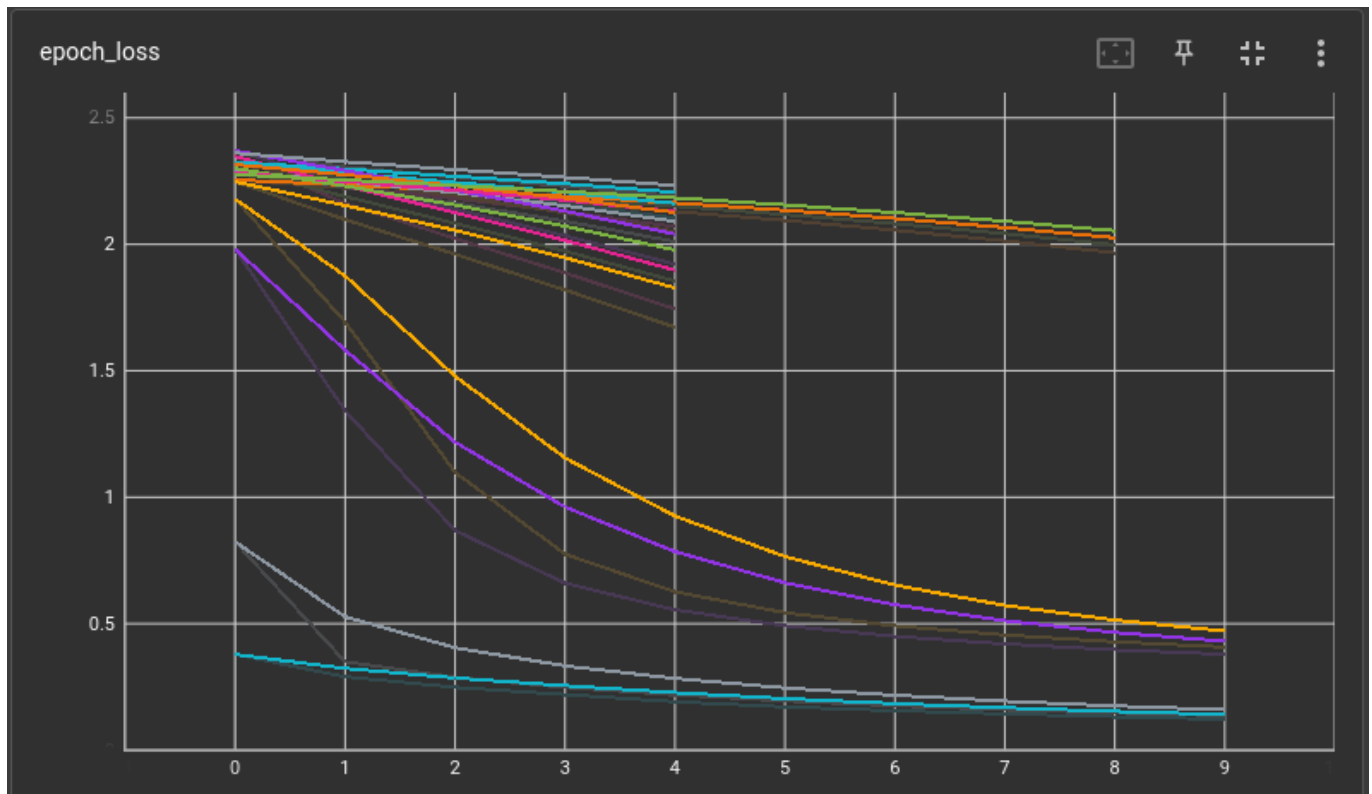
Hier kann man die erreichte Accuracy mit den standart Hyperparamter sehen.

```
938/938 [=====] - 37s 38ms/step - loss: 1.3486 - accuracy: 0.5960 - val_loss: 0.3780 - val_accuracy: 0.8930
Epoch 2/10
938/938 [=====] - 28s 30ms/step - loss: 0.3732 - accuracy: 0.8902 - val_loss: 0.2902 - val_accuracy: 0.9172
Epoch 3/10
938/938 [=====] - 27s 29ms/step - loss: 0.2994 - accuracy: 0.9126 - val_loss: 0.2480 - val_accuracy: 0.9298
Epoch 4/10
938/938 [=====] - 27s 28ms/step - loss: 0.2546 - accuracy: 0.9251 - val_loss: 0.2194 - val_accuracy: 0.9369
Epoch 5/10
938/938 [=====] - 27s 28ms/step - loss: 0.2248 - accuracy: 0.9341 - val_loss: 0.1911 - val_accuracy: 0.9455
Epoch 6/10
938/938 [=====] - 27s 28ms/step - loss: 0.2033 - accuracy: 0.9414 - val_loss: 0.1707 - val_accuracy: 0.9512
Epoch 7/10
938/938 [=====] - 32s 34ms/step - loss: 0.1730 - accuracy: 0.9502 - val_loss: 0.1557 - val_accuracy: 0.9571
Epoch 8/10
938/938 [=====] - 27s 28ms/step - loss: 0.1643 - accuracy: 0.9526 - val_loss: 0.1431 - val_accuracy: 0.9615
Epoch 9/10
938/938 [=====] - 29s 31ms/step - loss: 0.1498 - accuracy: 0.9577 - val_loss: 0.1317 - val_accuracy: 0.9653
Epoch 10/10
938/938 [=====] - 28s 30ms/step - loss: 0.1374 - accuracy: 0.9611 - val_loss: 0.1218 - val_accuracy: 0.9670
[7 2 1 0 4]
[7 2 1 0 4]
```

d)

Hier kann man die TensorBoard Auwertung sehen.





e)

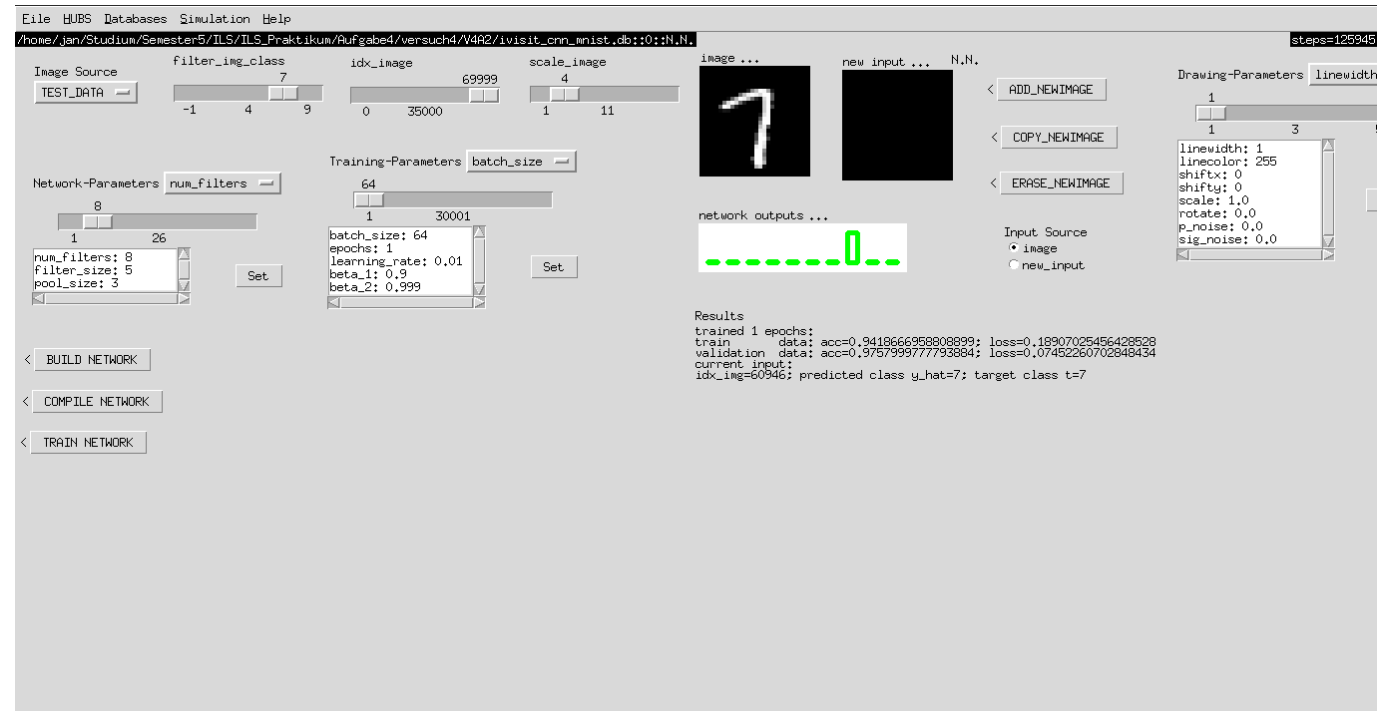
- Die größte Auswirkung auf die Accuracy hat die Lernrate.
- Für sehr kleine Mini-Batches ist die Modellberechnung sehr viel langsamer und damit konvergiert auch das Modell sehr viel langsamer
- Eine bessere Leistung als der einfache Stochastische Gradientenabstieg (SGD) ist der Adaptive Moment Estimation (ADAM). Er passt die Lernrate für jedes Gewicht separat an und berücksichtigt den ersten und auch den zweiten Moment der Gradienten.

## Aufgabe 2

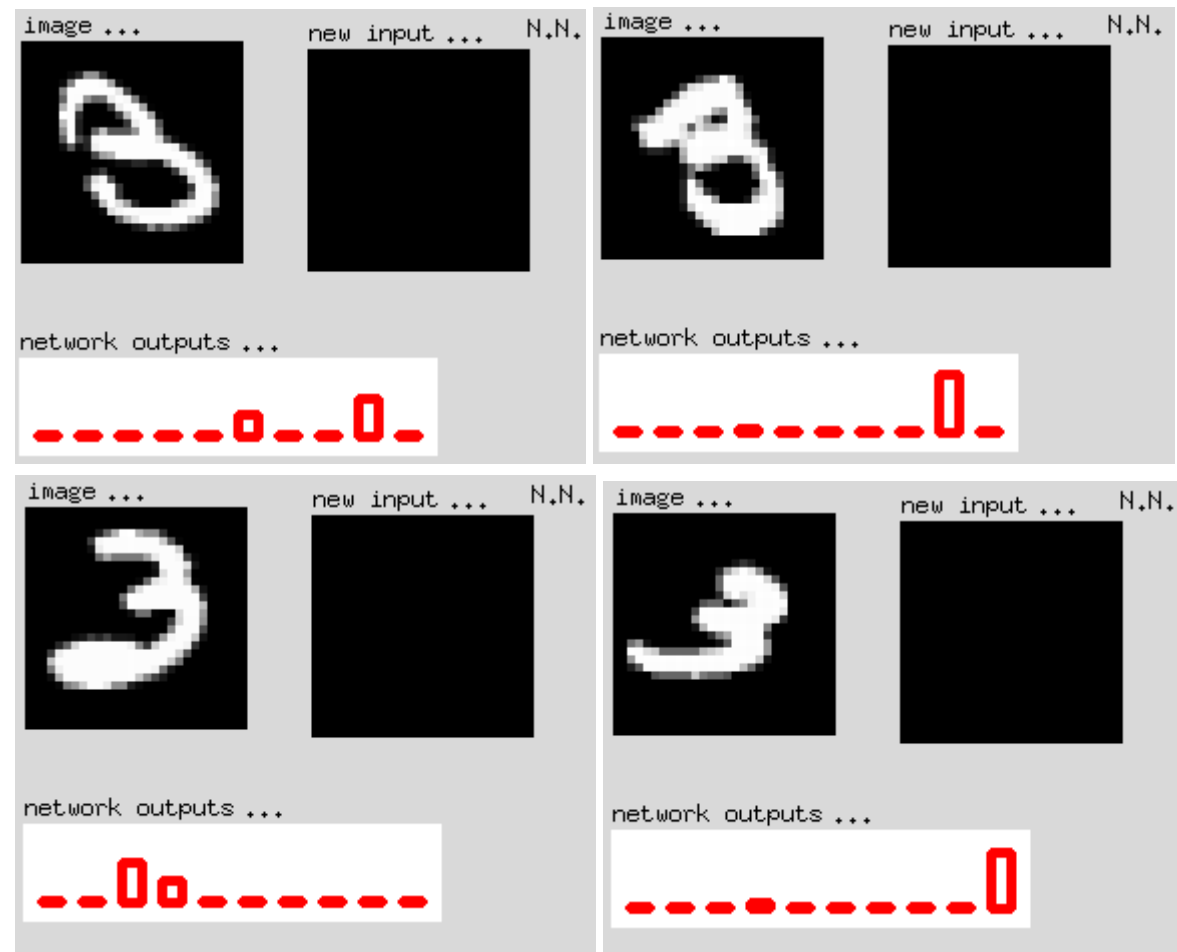
a)

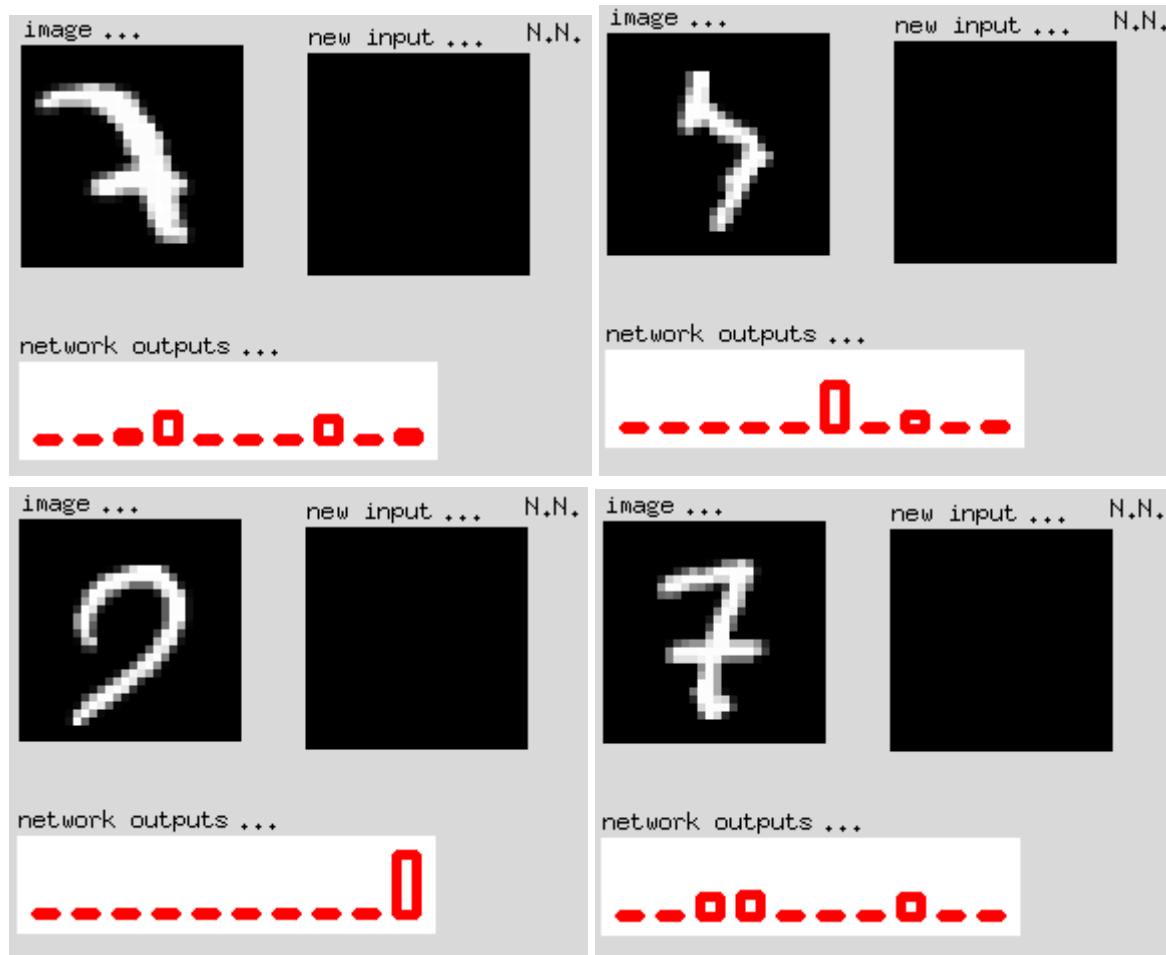
Es fällt zum Beispiel auf das bei der sieben auch eine Amerikanische Schreibweise vorhanden ist, bei der es keinen Querstrich gibt.

b)





c)





d)

image ...      new input ...      N.N.





< ADD\_NEWIMAGE

< COPY\_NEWIMAGE

< ERASE\_NEWIMAGE



network outputs ...



Input Source  
☐ image  
☒ new\_input

Results  
 trained 1 epochs:  
 train data: acc=0.9418666958808899; loss=0.18907025456428528  
 validation data: acc=0.9757999777793884; loss=0.07452260702848434  
 current input:  
 idx\_img=60855; predicted class y\_hat=2; target class t=3

image ...      new input ...      N.N.





< ADD\_NEWIMAGE

< COPY\_NEWIMAGE

< ERASE\_NEWIMAGE


network outputs ...




Input Source  
☐ image  
☒ new\_input

Results  
 trained 1 epochs:  
 train data: acc=0.9418666958808899; loss=0.18907025456428528  
 validation data: acc=0.9757999777793884; loss=0.07452260702848434  
 current input:  
 idx\_img=60855; predicted class y\_hat=3; target class t=3

image ...



new input ... N.N.




< ADD\_NEWIMAGE

< COPY\_NEWIMAGE

< ERASE\_NEWIMAGE

network outputs ...



Input Source

☐ image

☒ new\_input

Results

trained 1 epochs:


train data: acc=0.9418666958808899; loss=0.18907025456428528

validation data: acc=0.9757999777793884; loss=0.07452260702848434

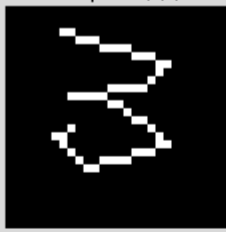
current input:

idx\_img=60855; predicted class y\_hat=3; target class t=3

image ...



new input ... N.N.




< ADD\_NEWIMAGE

< COPY\_NEWIMAGE

< ERASE\_NEWIMAGE

network outputs ...



Input Source

☐ image

☒ new\_input

Results

trained 1 epochs:


train data: acc=0.9418666958808899; loss=0.18907025456428528

validation data: acc=0.9757999777793884; loss=0.07452260702848434

current input:

idx\_img=60855; predicted class y\_hat=3; target class t=3

image ...



new input ... N.N.



< ADD\_NEWIMAGE

< COPY\_NEWIMAGE

< ERASE\_NEWIMAGE

network outputs ...

..00.....

Results

trained 1 epochs:  
train data: acc=0.9418666958808899; loss=0.18907025456428528  
validation data: acc=0.9757999777793884; loss=0.07452260702848434  
current input:  
idx\_img=60855; predicted class y\_hat=2; target class t=3

image ...



new input ... N.N.



< ADD\_NEWIMAGE

< COPY\_NEWIMAGE

< ERASE\_NEWIMAGE

network outputs ...


0.....

Results


trained 1 epochs:  
train data: acc=0.9418666958808899; loss=0.18907025456428528  
validation data: acc=0.9757999777793884; loss=0.07452260702848434  
current input:  
idx\_img=60855; predicted class y\_hat=0; target class t=3



image ...



new input ... N.N.




< ADD\_NEWIMAGE

< COPY\_NEWIMAGE

< ERASE\_NEWIMAGE

network outputs ...



Input Source

☐ image

☒ new\_input

Results

trained 1 epochs:

train data: acc=0.9418666958808899; loss=0.18907025456428528

validation data: acc=0.9757999777793884; loss=0.07452260702848434

current input:

idx\_img=60855; predicted class y\_hat=3; target class t=3

image ...



new input ... N.N.



< ADD\_NEWIMAGE

< COPY\_NEWIMAGE

< ERASE\_NEWIMAGE

network outputs ...



Input Source

☐ image

☒ new\_input

Results

trained 1 epochs:

train data: acc=0.9418666958808899; loss=0.18907025456428528

validation data: acc=0.9757999777793884; loss=0.07452260702848434

current input:

idx\_img=60855; predicted class y\_hat=3; target class t=3

image ...      new input ...      N.N.

network outputs ...

Input Source  
☐ image  
☒ new\_input

Results  
 trained 1 epochs:  
 train data: acc=0.9418666958808899; loss=0.18907025456428528  
 validation data: acc=0.9757999777793884; loss=0.07452260702848434  
 current input:  
 idx\_img=60855; predicted class y\_hat=3; target class t=3

image ...      new input ...      N.N.

network outputs ...


Input Source  
☐ image  
☒ new\_input


Results  
 trained 1 epochs:  
 train data: acc=0.9418666958808899; loss=0.18907025456428528  
 validation data: acc=0.9757999777793884; loss=0.07452260702848434  
 current input:  
 idx\_img=60855; predicted class y\_hat=2; target class t=3

Es wurde eine Accuracy von 60%

erreicht bei den selbst gezeichneten dreiern.

e)


image ...  


new input ... N.N.  


< ADD\_NEWIMAGE

< COPY\_NEWIMAGE

< ERASE\_NEWIMAGE

network outputs ...  


Input Source  

☐ image

☒ new\_input

Results

trained 1 epochs:  
train data: acc=0.9418666958808899; loss=0.18907025456428528  
validation data: acc=0.9757999777793884; loss=0.07452260702848434  
current input:  
idx\_img=60015; predicted class y\_hat=8; target class t=5


Drawing-Parameters rotate


24

-180 0 180

linewidth: 1  
linecolor: 255  
shiftx: 0  
shifty: 0  
scale: 1.0  
rotate: 24.0  
p\_noise: 0.0  
sig\_noise: 0.0

Set


image ...  


new input ... N.N.  


< ADD\_NEWIMAGE

< COPY\_NEWIMAGE

< ERASE\_NEWIMAGE

network outputs ...  


Input Source  

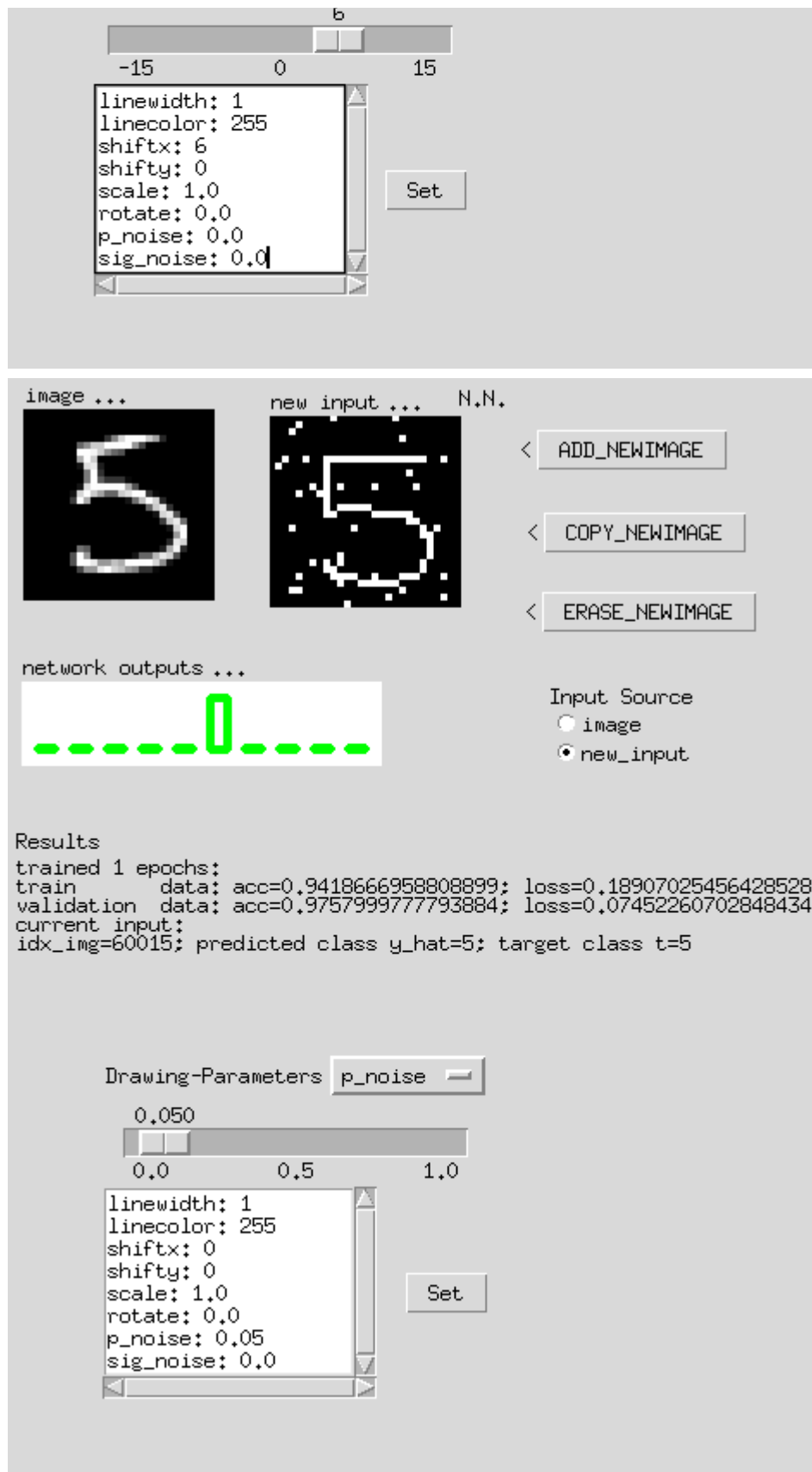
☐ image

☒ new\_input

Results

trained 1 epochs:  
train data: acc=0.9418666958808899; loss=0.18907025456428528  
validation data: acc=0.9757999777793884; loss=0.07452260702848434  
current input:  
idx\_img=60015; predicted class y\_hat=6; target class t=5

Drawing-Parameters shiftx



f)

Leider war es uns nicht möglich die fünf so zu manipulieren das eine sechs erkannt wurde. Meistens wurde eine 8 oder teilweise auch eine drei erkannt.

## Aufgabe 3

```

Epoch 8/10
250/250 [=====] - 151s 605ms/step - loss: 0.3458 - accuracy: 0.8785 - val_loss: 0.7958 - val_accuracy: 0.7512
Epoch 9/10
250/250 [=====] - 151s 605ms/step - loss: 0.2810 - accuracy: 0.8995 - val_loss: 0.8349 - val_accuracy: 0.7576
Epoch 10/10
250/250 [=====] - 152s 607ms/step - loss: 0.2225 - accuracy: 0.9216 - val_loss: 0.9236 - val_accuracy: 0.7535
1/1 [=====] - 0s 183ms/step
[[3 8 8 0 6]
 [[3]
  [8]
  [8]
  [0]
  [6]]

```

```

opt = Adam(learning_rate=0.001) # Optimierungsalgorithmus: Adam mit Lernrate 0.001
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])

# train model
log_dir = "logs/fit/cnn_cifar10_" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
history=model.fit(
    trainX,
    trainY,
    #batch_size=500,
    batch_size=200,
    #epochs=5,
    epochs=10,
    validation_data=(testX,testY),
    callbacks=[tensorboard_callback],

```

Als Optimierungsalgorithmus haben wir Adam mit Lernrate 0.001 verwendet und ein Accuracy von 92 % erreicht.