F7PMIAS2 MNIST

Klasifikační úloha

V rámci předmětu F7PMIAS2 - Analýza signálu II. byla zadána cvičná úloha na vytvoření neuronové sítě pro klasifikaci datasetu MNIST. Tento dokument slouží jako stručný report k vypracování úlohy a případné dodatečné informace k specifickým bodům v zadání.

Body 1-4

Většina zadání nevyžaduje dodatečný komentář, viz. zdrojový kód.

Figure 1: zobrazení datasetu

Trénovací set jsem ještě dále rozdělil na trénovací a validační podmnožinu se splitem 0,4.

```
training set: 60000 images
testing set: 10000 images
images per label:
label 0: 5923 images
label 1: 6742 images
label 2: 5958 images
label 3: 6131 images
label 4: 5842 images
label 5: 5421 images
label 6: 5918 images
label 7: 6265 images
label 8: 5851 images
label 9: 5949 images
train images size: (28, 28)
test images size: (28, 28)
```

Figure 2: výpisy informací o datasetu

Návrh architektury sítě

K návrhu architektury jsem přistupoval podle zmíněných metod v rámci přednášek. Použil jsem konvoluční vrstvy s různým počtem filtrů pro extrakci příznaků z dat, následně normalizoval a transformoval výstup pomocí flatten vrstvy pro výpočetní optimalizaci. Dále jsem použil dvě plně propojené vrstvy s dropout mezikrokem pro snížení overfittingu sítě.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320

batch_normalization (BatchN  (None, 26, 26, 32)        128
ormalization)

max_pooling2d (MaxPooling2D  (None, 13, 13, 32)        0
)

conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496

batch_normalization_1 (Batc  (None, 11, 11, 64)        256
hNormalization)

max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)          0
2D)

flatten (Flatten)            (None, 1600)              0

dense (Dense)                (None, 128)               204928

dropout (Dropout)            (None, 128)               0

dense_1 (Dense)              (None, 10)                1290

=================================================================
Total params: 225,418
Trainable params: 225,226
Non-trainable params: 192
```

Figure 3: shrnutí architektury sítě

Jako optimalizační metodu jsem zvolil Adaptive Moment Estimation. Zde jsem přesně neznal algoritmy běžně používaných metod, ale dle některých internetových zdrojů je Adam jednou z nejefektivnějších metod - např. proti běžnému gradientnímu sestupu. Jako loss funkci používám *sparse_categorical_crossentropy* pro klasifikaci do celočíselných tříd. Počet epoch jsem nastavil na 13.

Pro experimentování s parametry můžu zvažovat několik možností:

- počet epoch, možná i implementace early stoppingu
- trénovací batch size
- learning rate
- jiná metoda optimalizátoru
- změnit architekturu modelu
    - počet filtrů konvolučních vrstev
    - velikost Dense vrstvy
    - změnit dropout rate
- augmentace dat
    - může pomoci generalizovat řešený problém
    - úpravy snímků jako je rotace, šum, zoom, apod…

```
141/141 [==============================] - 3s 6ms/step - loss: 0.4411 - accuracy: 0.8721 - val_loss: 0.1130 - val_accuracy: 0.9668
Epoch 2/13
141/141 [==============================] - 1s 4ms/step - loss: 0.1288 - accuracy: 0.9620 - val_loss: 0.0681 - val_accuracy: 0.9793
Epoch 3/13
141/141 [==============================] - 1s 4ms/step - loss: 0.0896 - accuracy: 0.9736 - val_loss: 0.0560 - val_accuracy: 0.9829
Epoch 4/13
141/141 [==============================] - 1s 4ms/step - loss: 0.0701 - accuracy: 0.9786 - val_loss: 0.0522 - val_accuracy: 0.9845
Epoch 5/13
141/141 [==============================] - 1s 4ms/step - loss: 0.0559 - accuracy: 0.9835 - val_loss: 0.0501 - val_accuracy: 0.9857
Epoch 6/13
141/141 [==============================] - 1s 4ms/step - loss: 0.0498 - accuracy: 0.9843 - val_loss: 0.0406 - val_accuracy: 0.9885
Epoch 7/13
141/141 [==============================] - 1s 4ms/step - loss: 0.0394 - accuracy: 0.9880 - val_loss: 0.0397 - val_accuracy: 0.9888
Epoch 8/13
141/141 [==============================] - 1s 5ms/step - loss: 0.0346 - accuracy: 0.9889 - val_loss: 0.0417 - val_accuracy: 0.9889
Epoch 9/13
141/141 [==============================] - 1s 5ms/step - loss: 0.0311 - accuracy: 0.9901 - val_loss: 0.0407 - val_accuracy: 0.9885
Epoch 10/13
141/141 [==============================] - 1s 4ms/step - loss: 0.0279 - accuracy: 0.9907 - val_loss: 0.0450 - val_accuracy: 0.9880
Epoch 11/13
141/141 [==============================] - 1s 4ms/step - loss: 0.0228 - accuracy: 0.9921 - val_loss: 0.0475 - val_accuracy: 0.9872
Epoch 12/13
141/141 [==============================] - 1s 4ms/step - loss: 0.0259 - accuracy: 0.9919 - val_loss: 0.0385 - val_accuracy: 0.9891
Epoch 13/13
141/141 [==============================] - 1s 4ms/step - loss: 0.0230 - accuracy: 0.9930 - val_loss: 0.0393 - val_accuracy: 0.9893
313/313 [==============================] - 1s 2ms/step - loss: 0.0351 - accuracy: 0.9906
Test accuracy: 0.9906
313/313 [==============================] - 0s 1ms/step
1/1 [==============================] - 0s 44ms/step
```

Figure 4: průběh trénování

```
Predicted class for the randomly selected image is: 3
Softmax probability distribution:
Class 0: 0.0000
Class 1: 0.0000
Class 2: 0.0000
Class 3: 1.0000
Class 4: 0.0000
Class 5: 0.0000
Class 6: 0.0000
Class 7: 0.0000
Class 8: 0.0000
Class 9: 0.0000
```

Figure 5: použití modelu na testovací množině

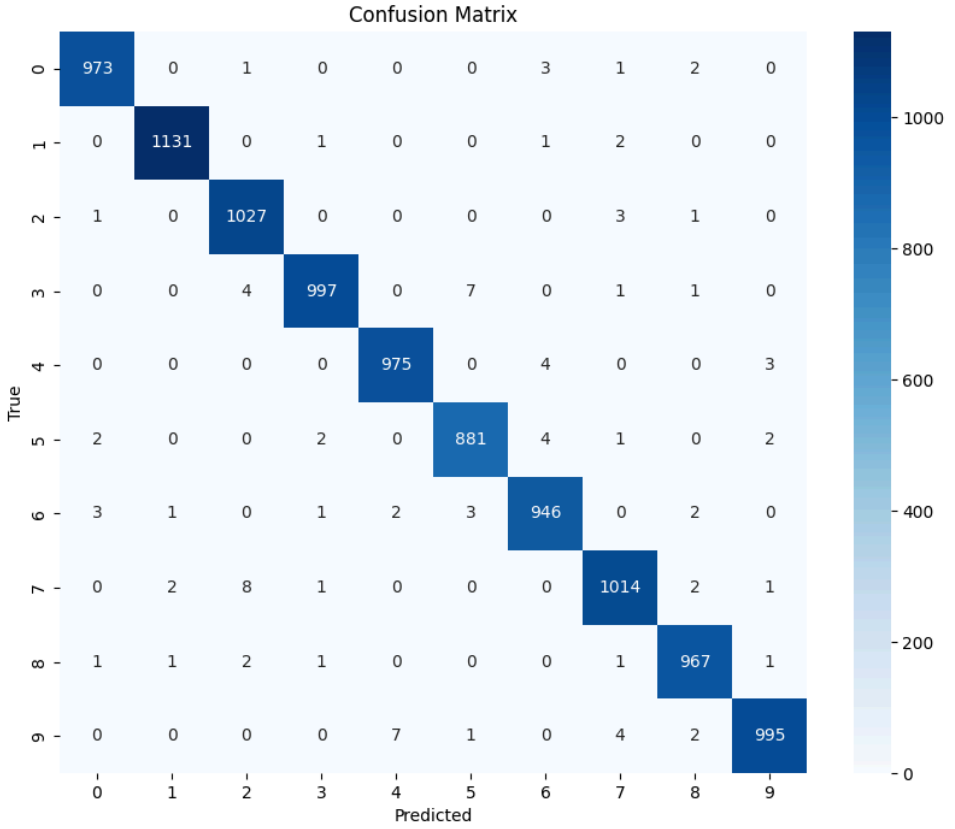

Figure 6: confusion matrix natrénovaného modelu na testovací množině

Závěr

V rámci zadané úlohy jsem navrhnul konvoluční neuronovou síť s deseti vrstvami a natrénoval ji s hyperparametry se kterými jsem dosáhl validační přesnosti 98,93 % v poslední epoše trénovacího procesu. Dokumentaci k jednotlivým bodům zadání jsem prezentoval především screenshoty terminálu nebo ploty z runtimu. Výsledná přesnost na testovací množině mi vyšla 99,06 %. Repozitář se zdrojovým kódem by měl být veřejně dostupný.