Implementing ANNs with TensorFlow

Session 13 - Practical Aspects

Agenda

1. Technical
    - Models
    - Sequential Definition
    - How to store and load a trained model?
2. What’s going on in Research?
    - Conferences
    - Important Institutions and Researchers
    - How to stay up to date?
    - Future Research
    - Courses

Model

TensorFlow Models

- In this course we sticked with defining a model by using the class `tf.keras.layers.Layer`

- In practice it often makes things easier to initialize a model as a `tf.keras.Model`.

tf.keras.Model

```
class Model(tf.keras.Model):

    def __init__(self):
        super(Model, self).__init__()
        self.conv_layer_1 = tf.keras.layers.Conv2D(
                                filters=32,
                                kernel_size=3,
                                activation=tf.keras.activations.relu,
                                input_shape=(32,32,3)
                            )
        self.max_pool_1 = tf.keras.layers.MaxPool2D()
        self.drop_out_1 = tf.keras.layers.Dropout(0.2)
        self.conv_layer_2 = tf.keras.layers.Conv2D(
                                filters=64,
                                kernel_size=3,
                                activation=tf.keras.activations.relu
                            )
        self.max_pool_2 = tf.keras.layers.MaxPool2D()
        self.flatten = tf.keras.layers.Flatten()
        self.fully_connected_1 = tf.keras.layers.Dense(
                                units=10,
                                activation=tf.keras.activations.softmax
                            )

    def call(self, x, training):
        x = self.conv_layer_1(x)
        x = self.max_pool_1(x)
        x = self.drop_out_1(x, training=training)
        x = self.conv_layer_2(x)
        x = self.max_pool_2(x)
        x = self.flatten(x)
        x = self.fully_connected_1(x)
        return x
```

Model class

Everything else as usual!

`is_training` flag should be just `training`

Compile the Model

```
model = Model()
optimizer = tf.keras.optimizers.Adam()
loss = tf.keras.losses.CategoricalCrossentropy()
accuracy = tf.keras.metrics.CategoricalAccuracy()
model.compile(optimizer,loss, metrics=[accuracy])
```

- **Initialize the model, optimizer, loss and metrics (e.g. accuracy).**
- **You have to make sure that the loss and the accuracy can be computed on your data (e.g. you have to create the one-hot labels before).**

Train the Model

```
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.cifar10.load_data()
train_labels = train_labels[:,0]
train_labels = tf.one_hot(train_labels, depth=10).numpy()
test_labels = test_labels[:,0]
test_labels = tf.one_hot(test_labels, depth=10).numpy()
train_images = np.array(train_images, dtype=np.float32)
test_images = np.array(test_images, dtype=np.float32)
train_images, test_images = train_images / 255.0, test_images / 255.0
```

You can simply feed the data as numpy arrays!

```
model.fit(train_images, train_labels, batch_size=32, epochs=3,  validation_data=(test_images, test_labels))
```

This single line will train the model on the corresponding training data for 3 epochs with a batch size of 32 and validate once a epoch on the validation data.

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/3
50000/50000 [==============================] - 44s 884us/sample - loss: 1.5393 - categorical_accuracy: 0.4370
- val_loss: 1.3364 - val_categorical_accuracy: 0.5153
Epoch 2/3
50000/50000 [==============================] - 45s 908us/sample - loss: 1.1922 - categorical_accuracy: 0.5755
- val_loss: 1.1555 - val_categorical_accuracy: 0.5940
Epoch 3/3
50000/50000 [==============================] - 43s 863us/sample - loss: 1.0579 - categorical_accuracy: 0.6252
- val_loss: 1.0234 - val_categorical_accuracy: 0.6406
```

Model Summary

You can even read out a summary of the model.

```
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_3 (Conv2D)            multiple                  896       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 multiple                  0         
_________________________________________________________________
dropout_1 (Dropout)          multiple                  0         
_________________________________________________________________
conv2d_4 (Conv2D)            multiple                  18496     
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 multiple                  0         
_________________________________________________________________
conv2d_5 (Conv2D)            multiple                  36928     
_________________________________________________________________
flatten_1 (Flatten)          multiple                  0         
_________________________________________________________________
dense_2 (Dense)              multiple                  65600     
_________________________________________________________________
dense_3 (Dense)              multiple                  650       
=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

Sequential Model Definition

- The sequential model definition makes things even easier.

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Conv2D(filters=32,
                                 kernel_size=3,
                                 activation=tf.keras.activations.relu,
                                 input_shape=(32,32,3)))
model.add(tf.keras.layers.MaxPool2D())
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Conv2D(filters=32,
                                 kernel_size=3,
                                 activation=tf.keras.activations.relu))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(units=10,
                                activation=tf.keras.activations.softmax))
```

- These lines are all you need for the definition of the model.

Saving a Model

- Saving a model is very easy:

```
model.save('stored_model.h5')
```

- This will store the weights plus the optimizer and its state.

- If you reload the model and train on you will start from the exact point that you stopped.

```
new_model = tf.keras.models.load_model('stored_model.h5')
```

What's happening in Research?

Conferences

- In opposite to most fields of research machine learning papers are mainly published on conferences.
- It can be quite interesting to visit conferences to get into touch with researchers and get a feeling for the field.
- The most important conferences are:
    - International Conference on Learning Representations (ICLR)
    - International Conference on Machine Learning (ICML)
    - Neural Information Processing Systems (NeurIPS)
    - AAAI Conference on Artificial Intelligence
    - Application oriented ones: e.g CVPR, ICCV, ACL

Conferences

- Sadly the most of these conference do not take place in Europe, but most of the time in the US or Canada.
- But ICML 2020 will be in Vienna!
- And in Tuebingen there will be the Machine Learning Summer School with some popular international speakers (http://mlss.tuebingen.mpg.de/2020/speakers.html).

NeurIPS Challenges

- The NeurIPS conference always features challenges on specific topics (e.g. Causality for Climate Change).
- Everyone can take part in these competitions and you usually only need to submit code/results and a small report. (not a complete publication).
- These can be a nice way to take part in a conference.
- Link

Open Review

- Sometimes conferences organize the reviewing process via https://openreview.net.

- ICLR does this every year.

- It can be interesting to read the reviews and the author's answers to get a feeling for how research is conducted.

Conference Statistics

- For most conferences you will find published statistics.
- These give you information about the institutions and authors and thus show you, where most research is happening.
- In the last years the contribution of industrial institutions increased rapidly.

Industrial Research

- Google (incl. Google Brain, Google Deep Mind)
- All other big tech companies: Microsoft, Facebook, IBM, Amazon
- Tons of other bigger companies e.g.: Bosch, Adobe, NVIDIA
- Smaller startup AI companies e.g.: CuriousAI, CriteoAI, Vicarious
- OpenAI: quite few but high quality publications

Academia

- Main academic players in U.S. and Canada:
  - MIT (Tomaso Poggio, Josh Tenenbaum)
  - Berkeley (California) (Sergey Levine, Pieter Abbeel)
  - Stanford (Li Fei-Fei, Chris Manning)
  - Carnegie Mellon (Russ Salakhutdinov)
  - Georgia Tech
  - Toronto (Vector Institute + Hinton)
  - Montreal (MILA + Bengio)

Academia - Europe

- GB: U. o. Oxford, U. o. Cambridge, U. o. Edinburgh, University College London, Imperial College London
- Switzerland: ETH Zuerich, EFPL Lausanne, IDSIA Lab (Schmidhuber)
- Germany: U. o. Tübingen (Schölkopf, Bethge), KIT (Karlsruhe), TU Darmstadt, U. o. Saarland (Saarbrücken)
- Others: Aalto University (Finland), KTH (Stockholm, Schweden), U. o. Amsterdam (Max Welling)

Get into it

- If you want to get into the field of research and see what is going on:
  - Follow the conferences and read important papers (spotlight).
  - Follow ArXiV. But be aware that anyone can upload papers here so it's very crowded and unreviewed.
  - Twitter (twitter-active researchers who might be a good starting point: @hardmaru, @rasbt, @jeremyphoward, @fchollet, @zacharylipton)
  - Online blogs/news: Machine learning mastery, Lil-log, Deep Mind, Deep Hunt, Hacker Noon

Courses

- Online courses: fast.ai, deeplearning.ai, Coursera
- Stanford NLP course (Chris Manning)
- Deep Mind Reinforcement Learning (David Silver)
- MIT Introduction to Deep Learning

Hot Topics

Artificial vs. Human Intelligence

- There is a very general discussion going on, whether deep learning alone is suitable to model actual human intelligence.
- Why? Because processing raw inputs in only achievable with deep learning, while many higher cognitive functions (e.g. counting, compositional understanding) seem easier to implement on a symbolic level.
- Advocates for Deep Learning: everyone who is doing deep learning including the “godfathers” : Y. Bengio, Y. LeCun, G. Hinton.
- Critiques from Cognitive Scientist: e.g. Gary Marcus (author of Rebooting AI), Brenden Lake

Resources

- Building Machines that Think and Learn like People (Lake et. al, Paper) (in general publications from B. Lake)
- DeepMind Comment on Building Machines that Think and Learn like People (link)
- Talk of Y. Bengio on lifting Deep Learning to the next level (link)
- Review Paper on reconciling deep learning and symbolic AI.

Human-Like Category Learning

- The way how humans learn categories differs widely from how artificial neural networks are trained on a dataset of different classes.

Neural Networks …

- Have access to all categories from the start of training.
- Don't need to deal with images, which do not belong to any known categories.
- Need thousands of samples to learn a category in a robust way.
- Can't explain the underlying concept of a category.

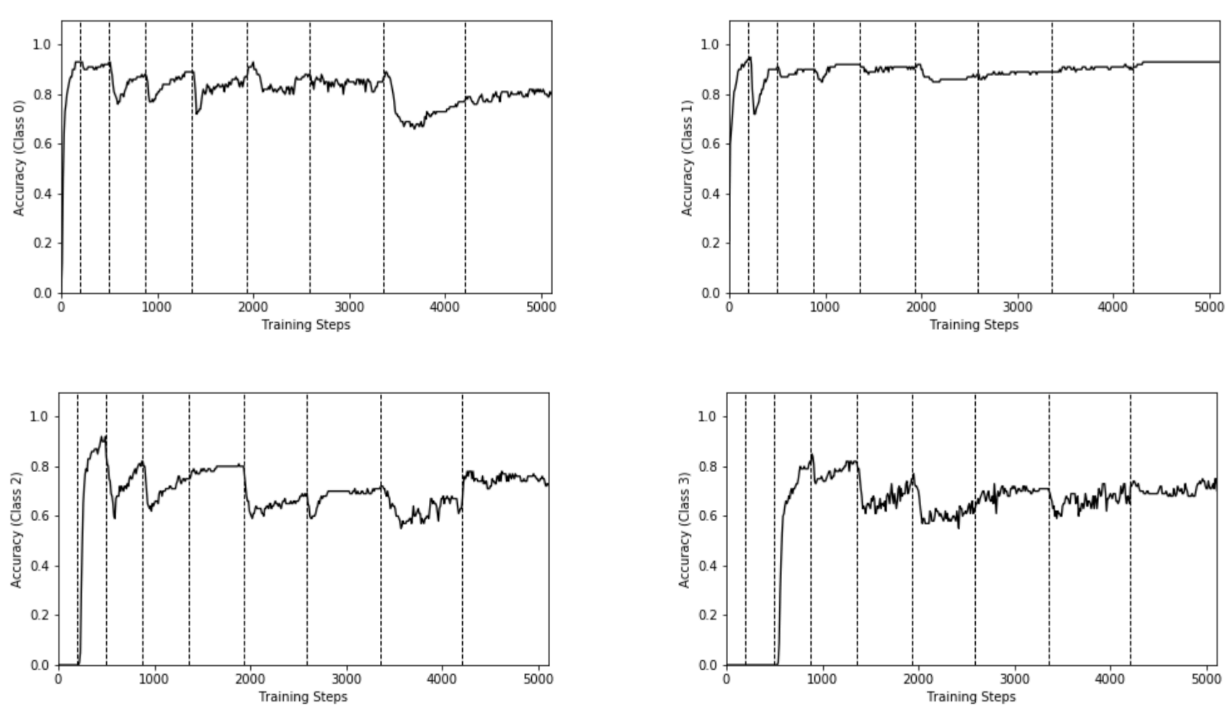
Humans …

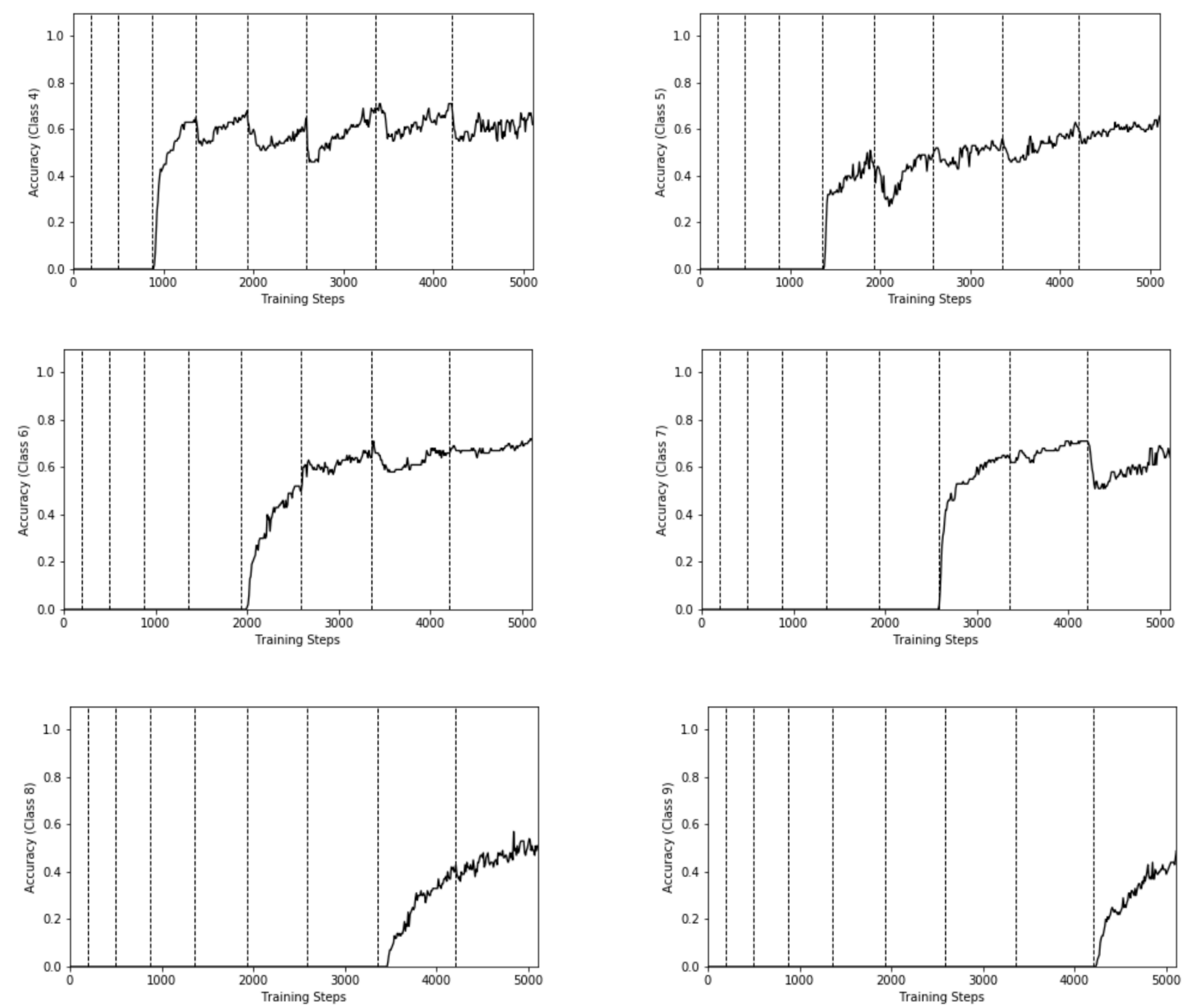
- Continually learn novel categories.
- Can detect when a sample is of an unknown category.
- Can learn categories from as few as one sample.
- Are somewhat aware of the underlying concept, which defines the category.

Continual Learning

- The task of learning categories one after the other is called **continual learning**.
- If neural networks learn this way one observes a phenomenon called **catastrophic forgetting**, which means that a network gets worst in recognizing “old” categories after it learned new categories.

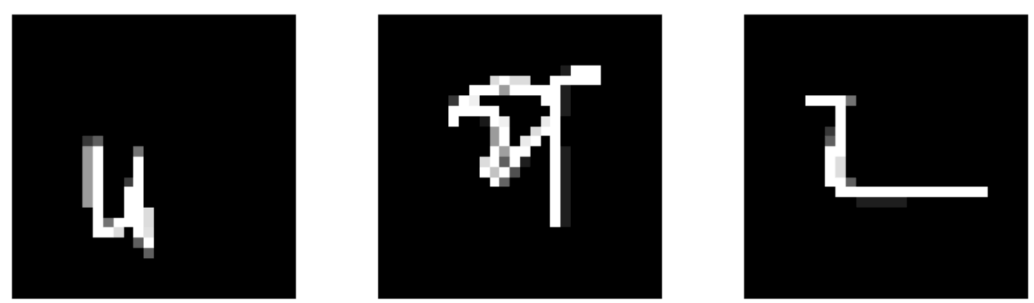
<u>Simple Example on MNIST</u>



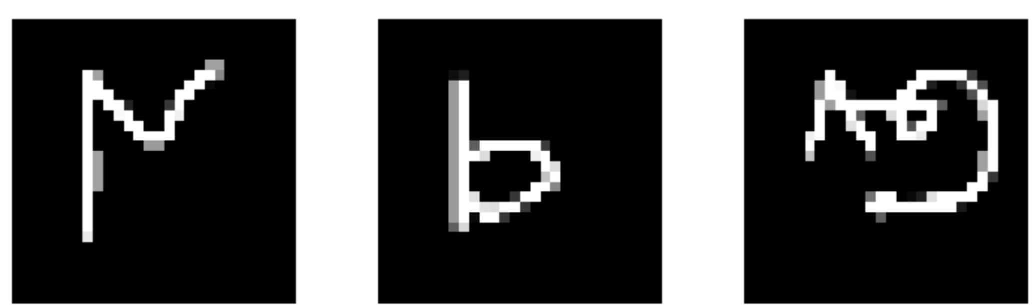


Out-of-Distribution Detection

- The task of understanding when a sample belongs to a category which is unknown is called **out-of-distribution detection** (OODD).
- Neural networks usually do not have this ability and sometimes assign high probabilities to out-of-distribution samples:



(a) Prediction: 6 Confidence: 0.92
(b) Prediction: 9 Confidence: 0.92
(c) Prediction: 4 Confidence: 0.96



(d) Prediction: 9 Confidence: 0.45
(e) Prediction: 6 Confidence: 0.72
(f) Prediction: 9 Confidence: 0.77

One-/Few-Shot Learning

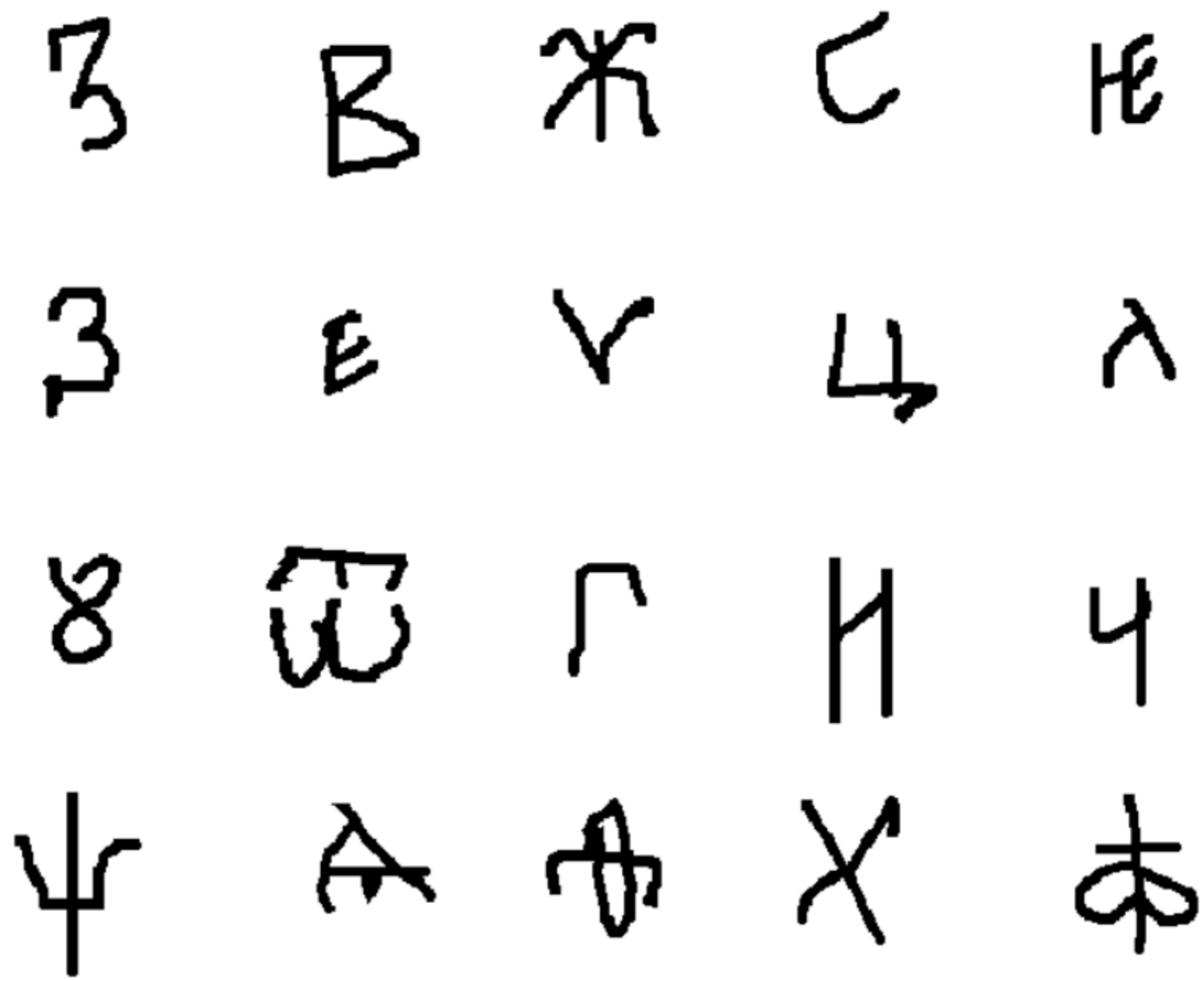
- The task of learning a new category from a few samples is called **one-** or **few-shot learning**.
- Humans are quite good at it while a classical neural networks performance drops heavily.
- A popular task to test on this is a so called **k-shot-view task**.

Example: 1-shot-20-view task

Try to learn/remember the following character!

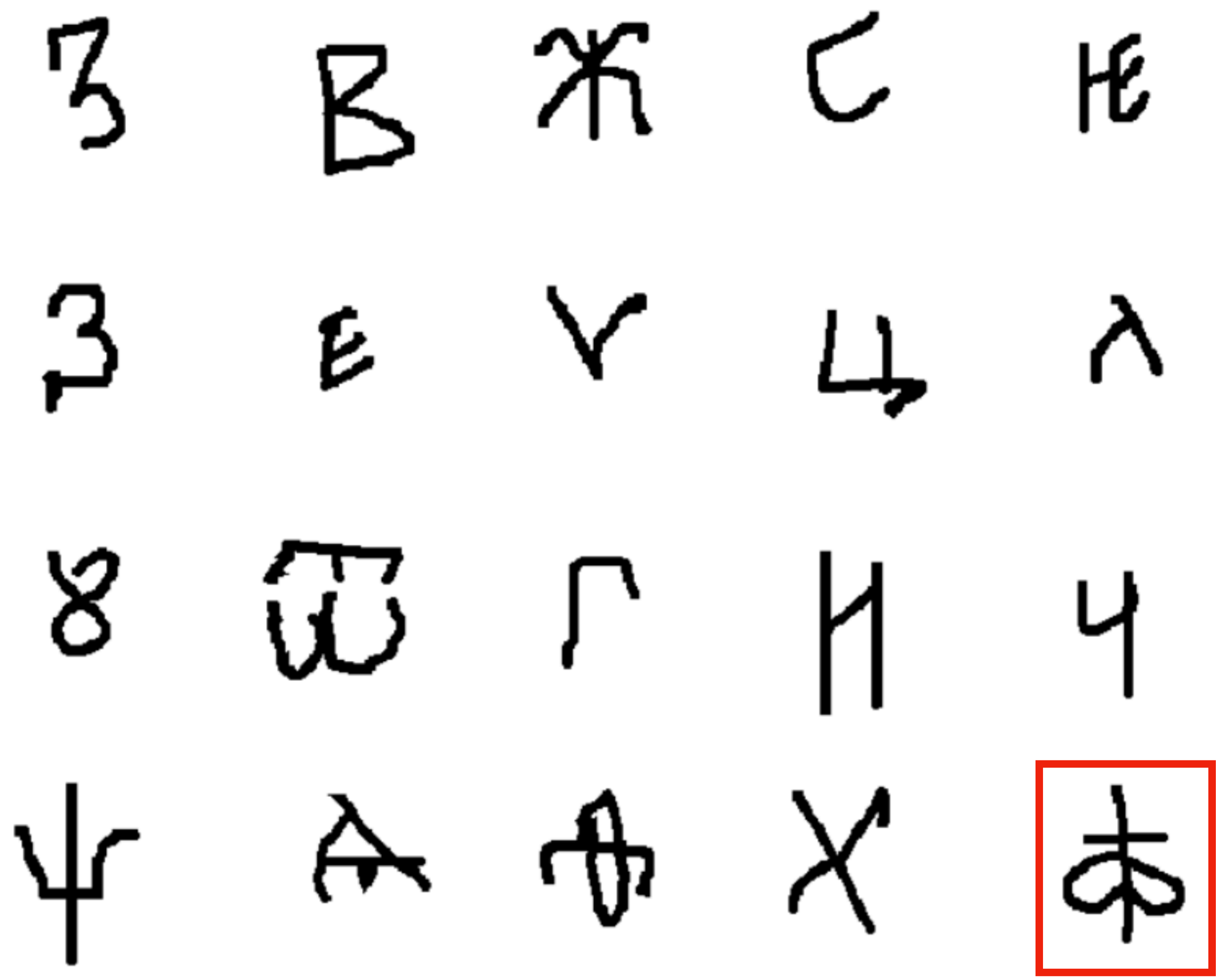
One-/Few-Shot Learning

Which of the following is the same character?



One-/Few-Shot Learning

Which of the following is the same character?



Omniglot Challenge

[Paper](#)

- The 1-shot-20-view task was presented as part of the **Omniglot Challenge**, a set of tasks designed to mirror human-level concept learning abilities.

Not available due to copyright issues.

i) One-shot learning (classification)

ii) One-shot learning (generation)

iii) Concept decomposition

iv) Generation of new concepts

Omniglot Challenge

- Since the publication of the challenge a lot of research was done on one-/few-shot learning (quite successfully for simple characters).
- Some work was done on one-shot generation, but very few was achieved on the decomposition part.
- See review paper from the authors of the omniglot challenge for a comprehensive summary of all relevant approaches.

Compositional Understanding

- The ability to decompose a concept touches on a very general notion of **compositional understanding**.
- Compositionality is an extremely important feature of human language (the meaning of a sentence is determined by its structure and the meaning of its words).
- Neural networks fail to leverage this aspect in their learning process.

Compositional Understanding

Paper

- Example Task for few-shot learning of compositional instructions:

Not available due to copyright issues.

Explainable AI

- The task of decomposing a concept into its "components" also touches on the field of **explainable AI**.
- Explainable AI is an important line of research working on ways of extracting human-interpretable rules from machine/deep learning algorithms.

Safety and Ethics in AI

- Explainable AI itself is part of a broader line of research summarized under **Safety of AI** (Stanford Research Center).
- This includes AI ethics (e.g. autonomous driving/ weapons), building trustworthy AI, building robust AI.
- This field is extremely important and will presumably get more and more important over the next years.

Other Important Fields

- Reinforcement Learning (incl. for robotics) - lecture 12
- Unsupervised Learning of Disentangled Representations - lecture 11

Final Project

Goal

- The goal of the final project is to test whether you can:
  - read a research paper,
  - abstract and comprehend the most important aspects,
  - implement the architecture and experiment in TensorFlow
  - and give a clear explanation of what you did.
- Of course it is nice to have an exciting project with awesome results, but this should not be your focus.

Submission Format

- The submission has to contain **code** and **written text**, explaining your project.
- The written text should contain:
    - Introduction (+relevant background knowledge, incl. citations).
    - Main part. Explaining the publication (architecture, loss, experiments…) accompanied with the important parts of the code.
    - Results + Visualization + Discussion
- **Choice 1**: One Jupyter notebook. With the written text in markdown and the code in the cells in between.
- **Choice 2**: Submit code via a GitHub repo. And write a pdf-report (using LaTeX). In this case include screenshots of important parts of the code.

Assessment

- The assessment of the project will be based on:
  - The displayed understanding in the report
  - Readability of report/code (include comments)
  - Clear visualization of experimental results.
  - Effectiveness of implementation.

It is fine to look for implementations in the web and get help and inspiration but be aware that we will scan for plagiarism, which will result in a FAIL!

Grid

- You can use the grid system for your final project.
- This means you can use computers located in the institute, which have GPUs.
- Working on the grid can be annoying though.
- If you would like to work on the grid please send me a mail until the end of this week.
- There is a Stud.IP group “Grid-Computing at the …”, where you can find infos on how to use it.

Last Words

Feedback

- The team and I would love to get some feedback on the course.
- Please fill out the following Google Form.

Goodbye!