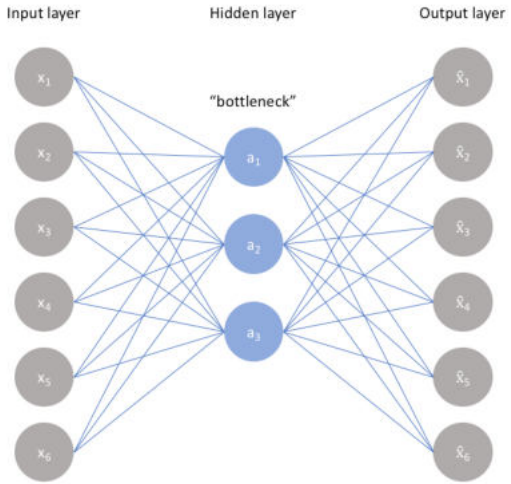
Introduction to Deep Learning (I2DL)

Exercise 8: Autoencoder

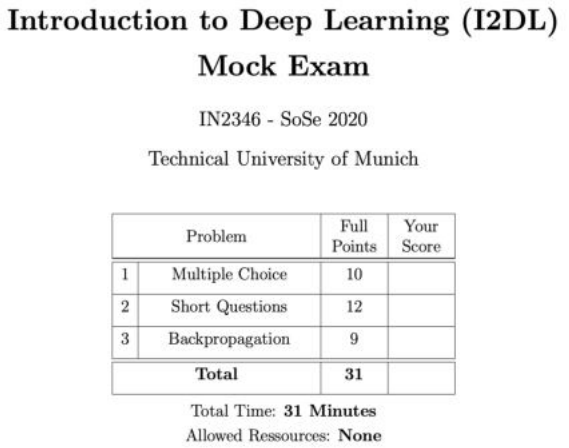
Today's Outline

- Exam
  - Mock Exam
- Hyperparameter tuning
- Exercise 8
  - Batch Normalization & Dropout
  - Transfer Learning
  - Autoencoder
- Personal: Github/Exposure



Mock Exam & Update

- Exam Structure via Mock Exam
  - Multiple Choice
  - Written Questions
  - No coding



Introduction to Deep Learning (I2DL)

Mock Exam

IN2346 - SoSe 2020

Technical University of Munich

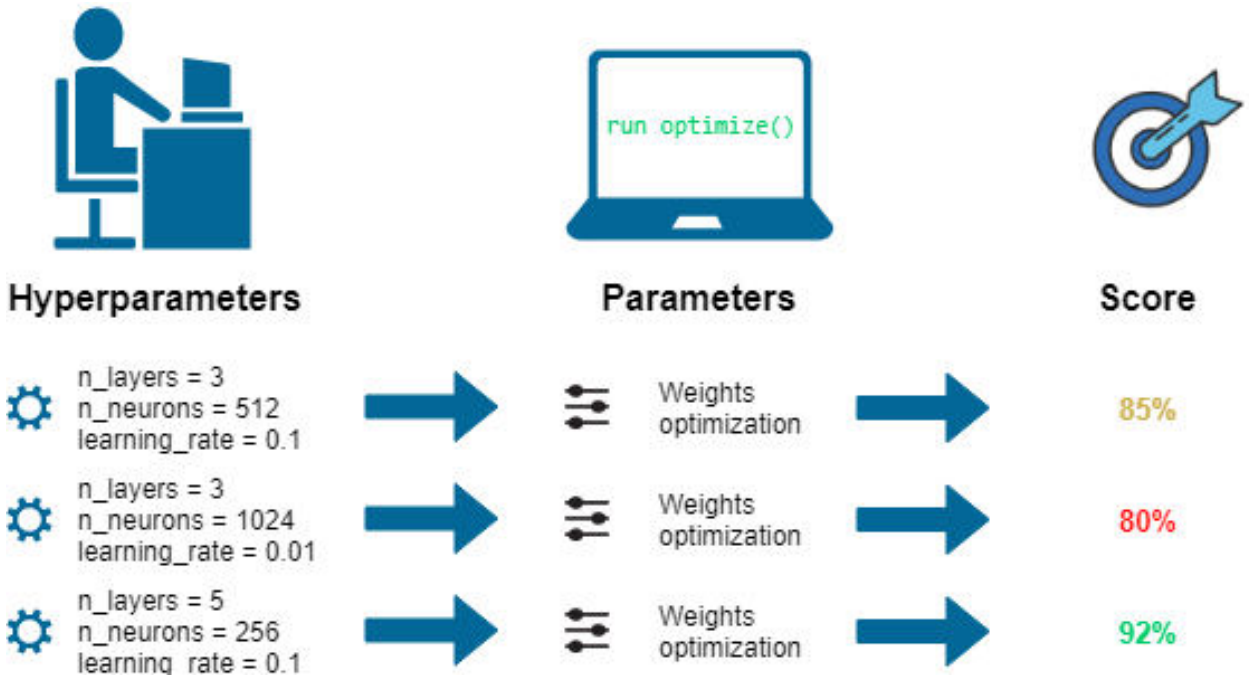
	Problem	Full Points	Your Score
1	Multiple Choice	10	
2	Short Questions	12	
3	Backpropagation	9	
	Total	**31**	

Total Time: **31 Minutes**

Allowed Ressources: **None**

- Exam
  - No concrete information from the university
  - We will keep you posted
  - Additional questions after CNNs and once we have more info

Hyperparameter Tuning



Source: https://images.deepai.org/glossary-terms/05c646fe1676490aa0b8cab0732a02b2/hyperparams.png

Hyperparameter Tuning

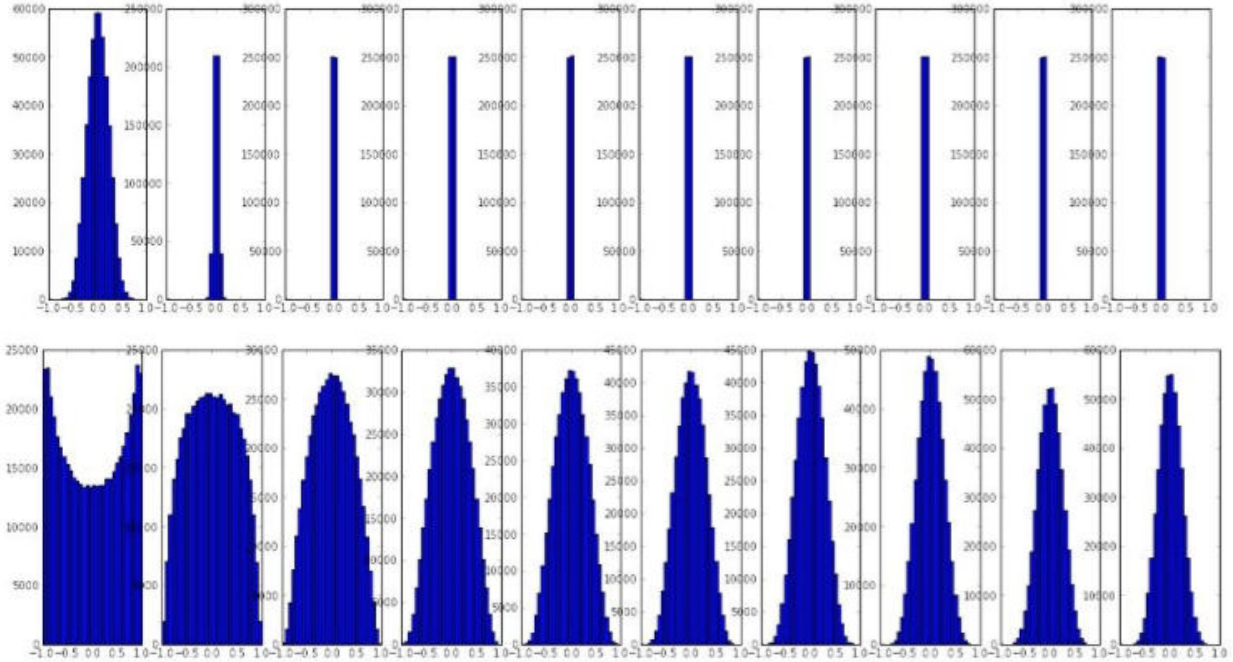
- Slides on Piazza
  - Check them out if you haven't done it yet

- It is important
  - Regardless of your resources
  - There is no all in one recipe
  - If you need more practice: optional submission on CIFAR10 (Solutions will be discussed next week)

Improve your training!

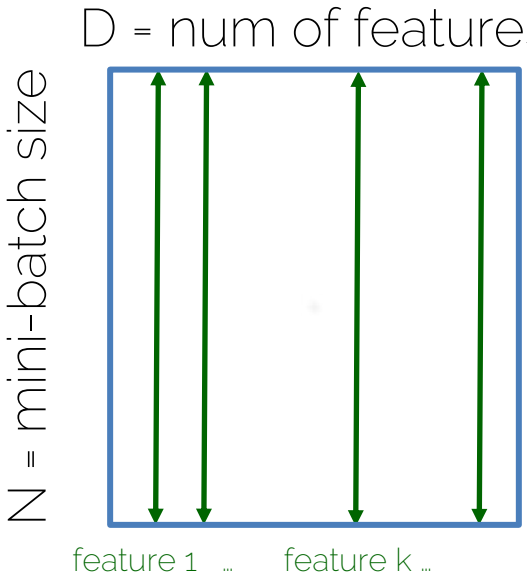
Batch Normalization

- All we want is that our activations do not die out



Batch Normalization

- Wish: Unit Gaussian activations



Mean of your mini-batch examples over feature k

$$\hat{\boldsymbol{x}}^{(k)} = \frac{\boldsymbol{x}^{(k)} - E[\boldsymbol{x}^{(k)}]}{\sqrt{Var[\boldsymbol{x}^{(k)}]}}$$

Unit gaussian

Batch Normalization

- 1. Normalize

$$\hat{\boldsymbol{x}}^{(k)} = \frac{\boldsymbol{x}^{(k)} - E[\boldsymbol{x}^{(k)}]}{\sqrt{Var[\boldsymbol{x}^{(k)}]}}$$

- 2. Allow the network to change the range

$$\boldsymbol{y}^{(k)} = \gamma^{(k)}\hat{\boldsymbol{x}}^{(k)} + \beta^{(k)}$$

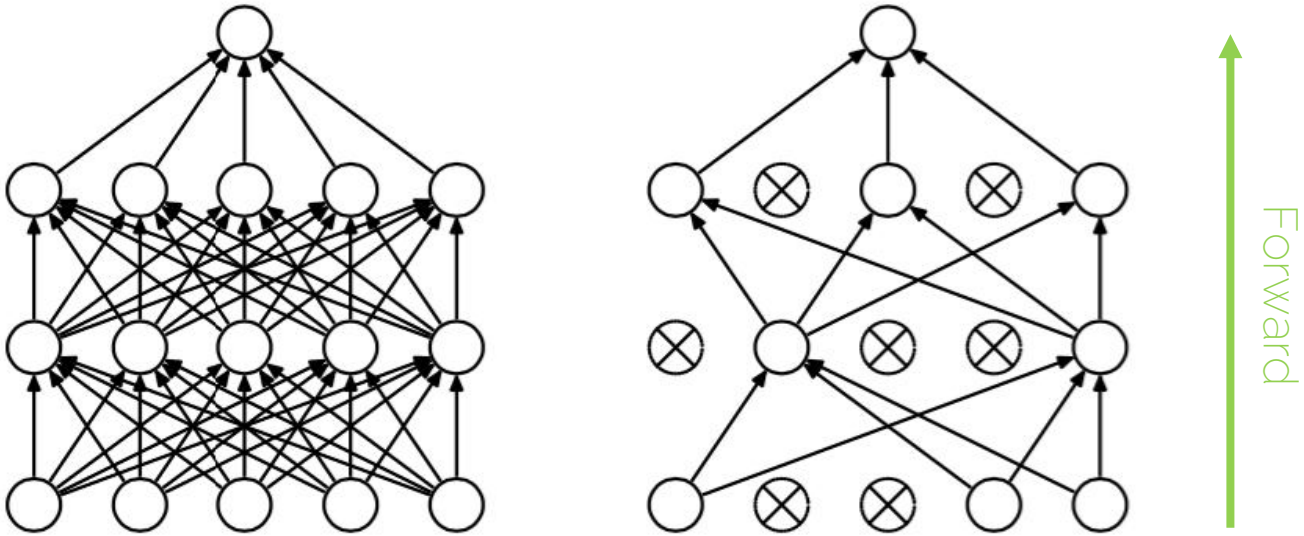
backprop

The network *can* learn to undo the normalization

$$\gamma^{(k)} = \sqrt{Var[\boldsymbol{x}^{(k)}]}$$

$$\beta^{(k)} = E[\boldsymbol{x}^{(k)}]$$

Dropout



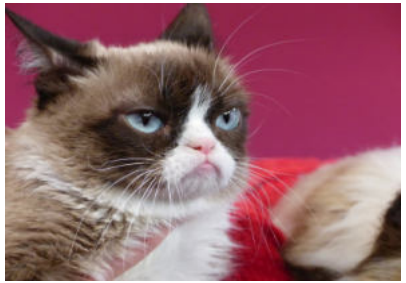
(a) Standard Neural Net

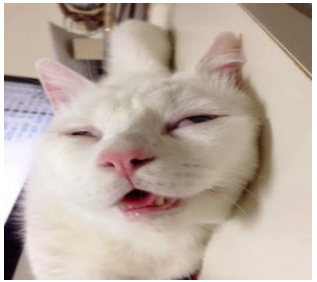
(b) After applying dropout.

- Using half the network = half capacity

Transfer Learning

Transfer Learning: Example Scenario



- Need to build a Cat classifier
- Only have a few images ~10 000

Transfer Learning

- Problem Statement:
  - Training a Deep Neural Network needs a lot of data
  - Collecting much data is expensive or just not possible

- Idea:
  - Some problems/ tasks are closely related
  - Can we transfer knowledge from one task to another?
  - Can we re-use (at least parts of) a pre-trained network for the new task?

Transfer Learning

Distribution

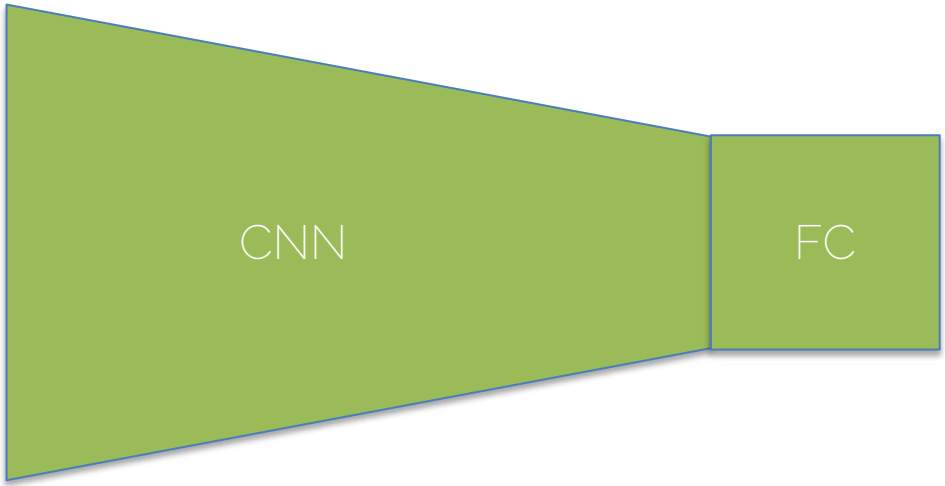
Large dataset

Distribution

Small dataset

Use what has been learned for another setting

Transfer Learning

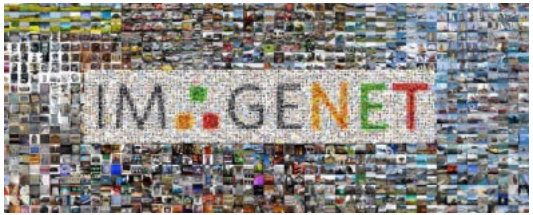


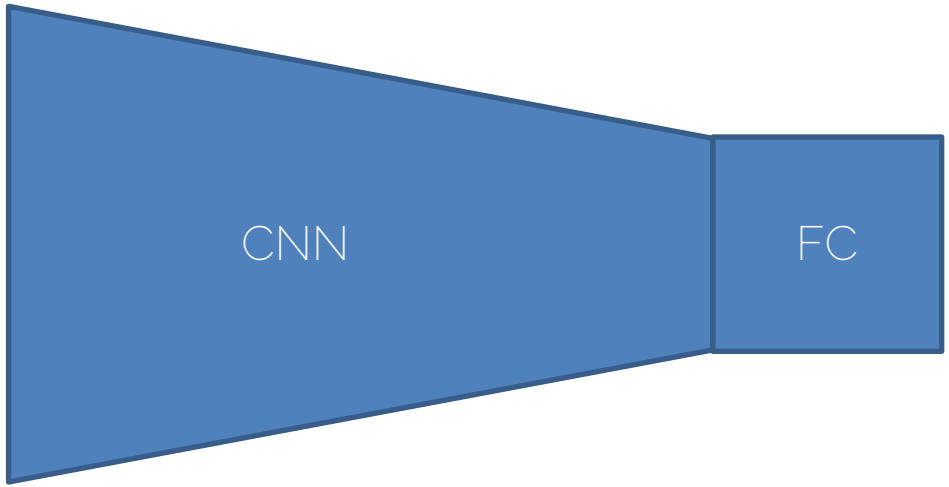
Coloring Legend:

Untrained

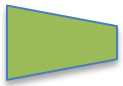
Trained

Transfer Learning





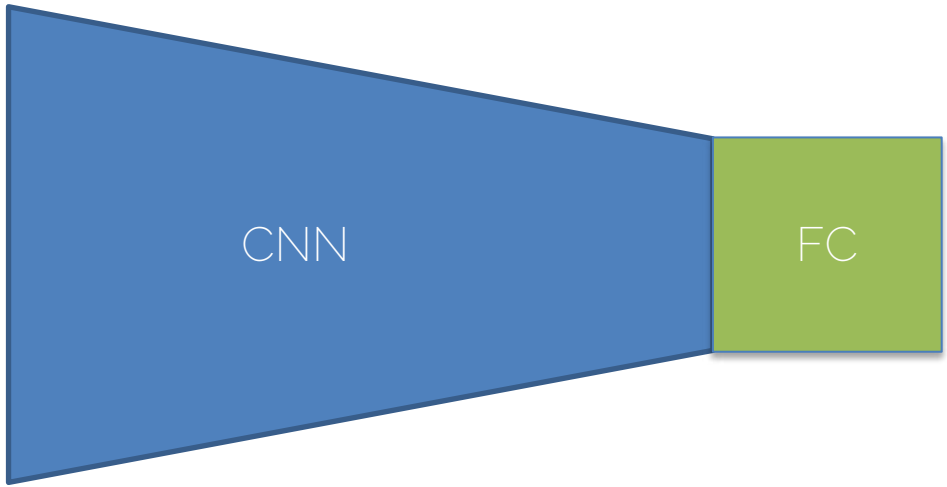
Coloring Legend:



Untrained

Trained

Transfer Learning



Coloring Legend:

Untrained

Trained

Feature Extractor

Classifier

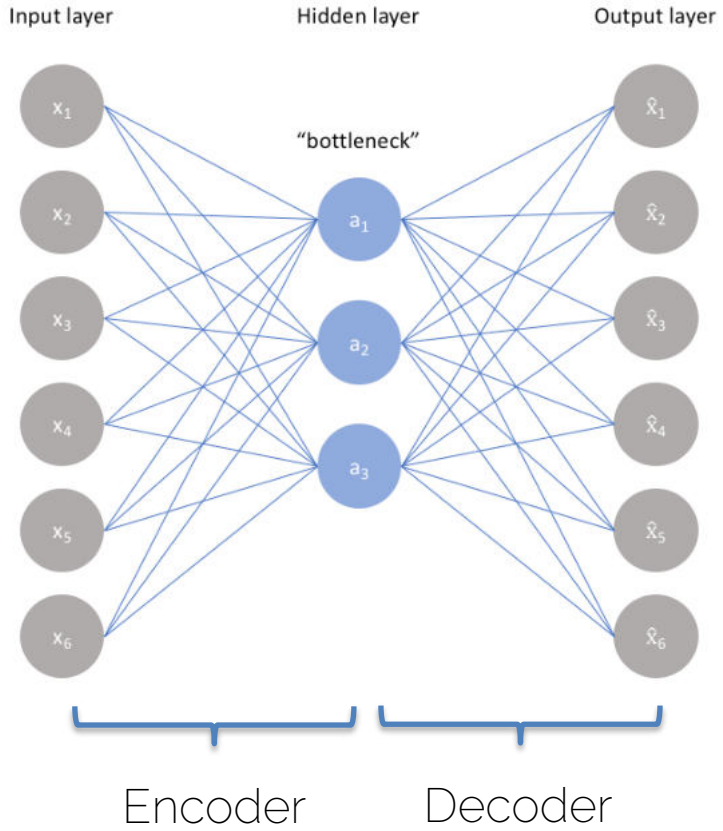
Maybe freeze weights/ slower learning rate/ nothing special

Newly initialized head

Application: Autoencoder (Sub 8)

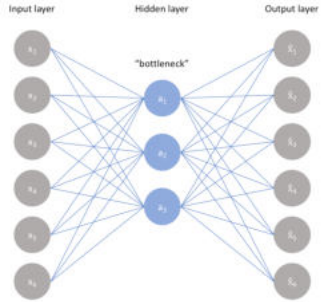
Autoencoder

- Task
  - Reconstruct the input given a lower dimensional bottleneck
  - Loss: L1/L2 per pixel
- Actually need no labels!
- Without non-linearities: similar to PCA

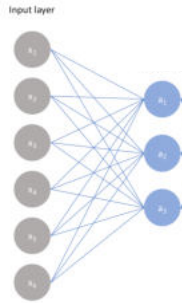


Transfer Using an Autoencoder

- Step 1:
  - Train an Autoencoder on a large (maybe unlabeled) dataset very similar to your target dataset



- Step 2:
  - Take pre-trained Autoencoder and use it as the first part of a classification architecture for your target dataset



Personal Note: Github/Exposure

- Posting I2DL solutions is not a helpful git for you
  - Maybe among other students…
- What is useful?
  - Something to talk about in interviews
- Projects:
  - internships/ guided research/ any task basically
  - Document your process (blog), show and visualize your data processing, discuss design decisions and publish code

See you next week!