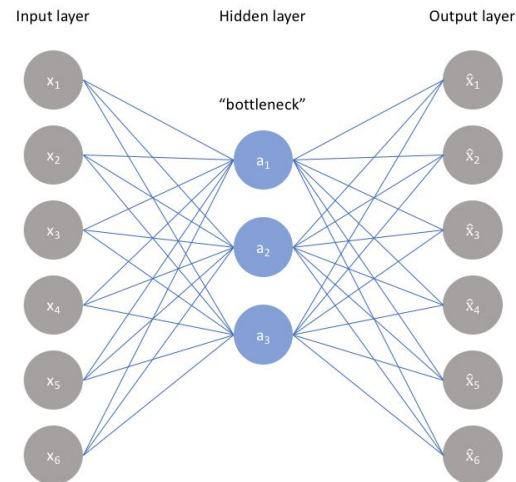# Introduction to Deep Learning (I2DL)
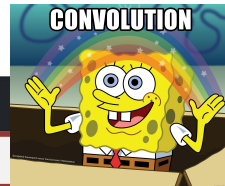
## Exercise 8: Autoencoder

# Today's Outline

- Exercise 07: Example Solutions
- Exercise 08
  - Batch Normalization & Dropout
  - Transfer Learning
  - Autoencoder

# Exercise 7: Solutions

# Leaderboard: Ex7



| # | User | Score |
|---|------|-------|
| 1 | u0787 | 64.30 |
| 2 | u0120 | 59.87 |
| 3 | u0807 | 56.85 |
| 4 | u0146 | 56.59 |
| 5 | u0746 | 55.47 |
| 6 | u0638 | 55.40 |
| 7 | u0766 | 54.34 |
| 8 | u0676 | 54.19 |
| 9 | u0853 | 54.16 |
| 10 | u1490 | 54.13 |

# Solution 1: 59,87%

Manual Transforms:
- Crop
- Gaussian filter
- Rotation
- Flip
- etc

```python
self.model = nn.Sequential(
    nn.Linear(self.hparams["input_size"], self.hparams["nn_hidden_Layer1"]),
    nn.ReLU(),
    nn.Linear(self.hparams["nn_hidden_Layer1"], self.hparams["num_classes"]),
    nn.ReLU()
    )
```

```python
my_transform = transforms.Compose([
                    transforms.ToTensor(),
                    transforms.Normalize(mean, std)])
```

```python
split = {
    'train': 0.9,
    'val': 0.05,
    'test': 0.05
}
split_values = [v for k,v in split.items()]
assert sum(split_values) == 1.0
```

```python
def configure_optimizers(self):

    optim = None
    ###############################################################
    # TODO: Define your optimizer.                               #
    ###############################################################

    optim = torch.optim.Adam(self.model.parameters(), self.hparams["learning_rate"], weight_decay=self.hparams['weight_decay'])
    StepLR = torch.optim.lr_scheduler.MultiStepLR(optim, milestones=[30],gamma=0.5)

    ###############################################################
    #                    END OF YOUR CODE                        #
    ###############################################################
    return optim
```

```python
hparams["loading_method"] = 'Memory'
hparams['num_workers'] = 1
hparams['input_size'] = 3 * 32 * 32
hparams['batch_size'] = 1000
hparams['learning_rate'] = 5e-5
hparams['weight_decay'] = 1e-3
hparams['nn_hidden_Layer1'] = 1500
hparams['num_classes'] = 10
```

I2DL: Prof. Dai

# Solution 2: 56,85%

```python
self.model = nn.Sequential(
    nn.Linear(self.hparams["input_size"], self.hparams["hidden_size"]),
    nn.ReLU(),
    nn.Linear(self.hparams["hidden_size"], self.hparams["num_classes"]),
    )
```

```python
my_transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize(mean, std), transforms.RandomCrop(32, padding=4),
transforms.RandomHorizontalFlip()])
```

```python
def configure_optimizers(self):

    optim = None
    ########################################################################
    # TODO: Define your optimizer.                                         #
    ########################################################################
    optim = torch.optim.SGD(self.model.parameters(), self.hparams["learning_rate"], momentum=0.9)
```

```python
# Note: you can change the splits if you want :)
split = {
    'train': 0.6,
    'val': 0.2,
    'test': 0.2,
}
split_values = [v for k,v in split.items()]
assert sum(split_values) == 1.0
```

```python
hparams = {
    "batch_size": 16,
    "learning_rate": 1e-3,
    "input_size": 3 * 32 * 32,
    "hidden_size": 512,
    "num_classes": 10,
    "num_workers": 2,    # used
}
```
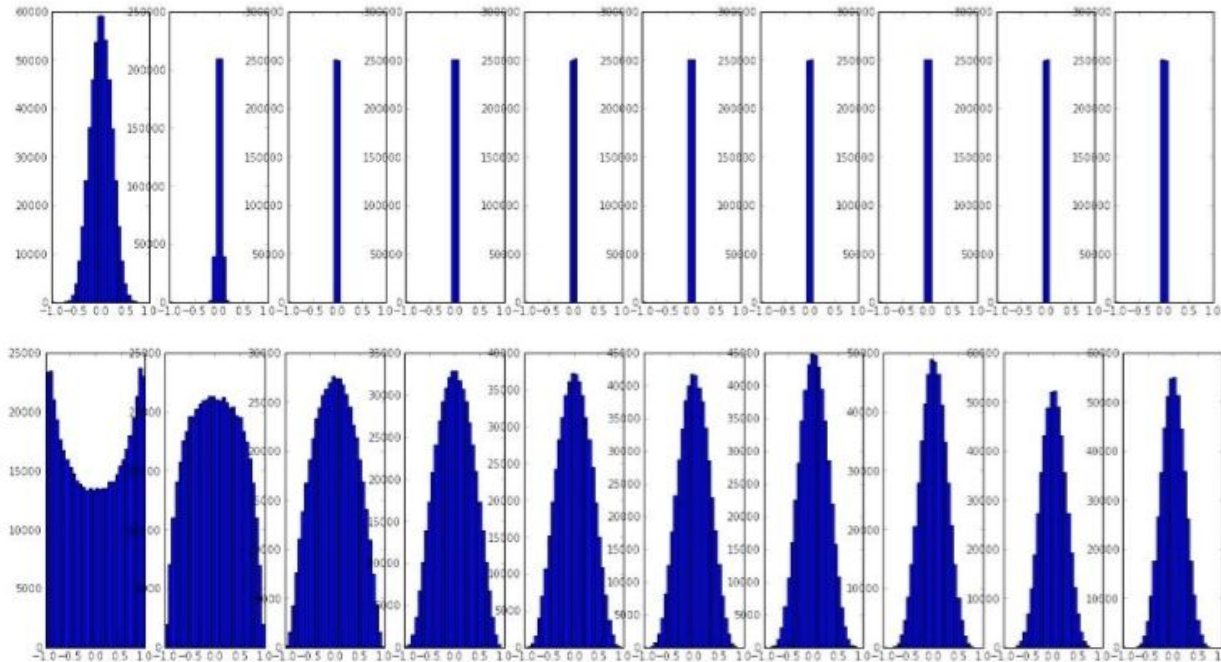
# Summary

- Network: Linear + ReLU (Depth: 2-4)
- Initialization of Network Weights
- Optimizer: SGD or Adam, LR Scheduler
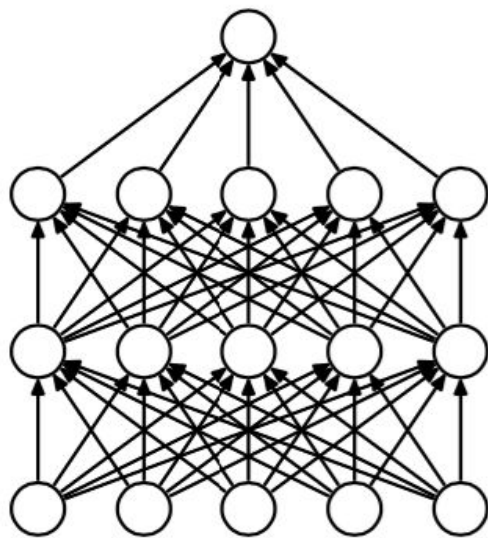- Data Augmentation

# Improve your training!

# Batch Normalization

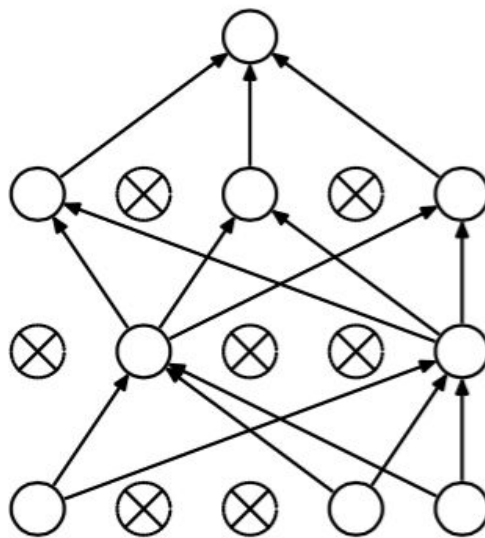- All we want is that our activations do not die out

# Dropout

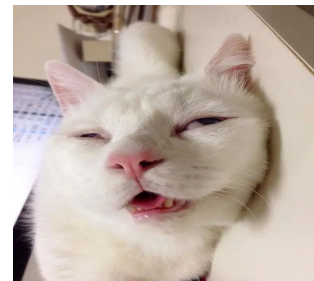- Using half the network = half capacity



(a) Standard Neural Net

(b) After applying dropout.

Forward

# Transfer Learning
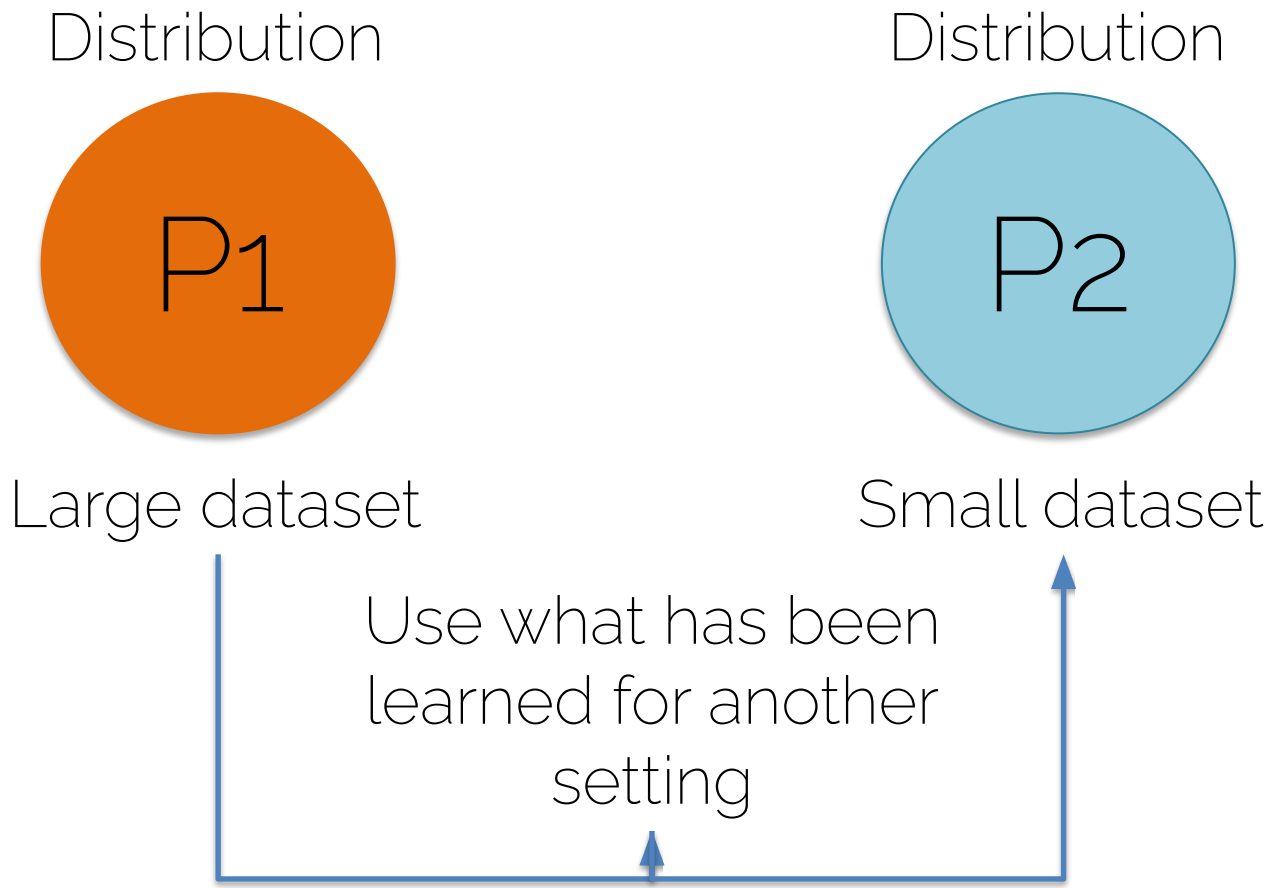
# Transfer Learning: Example Scenario



- 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?
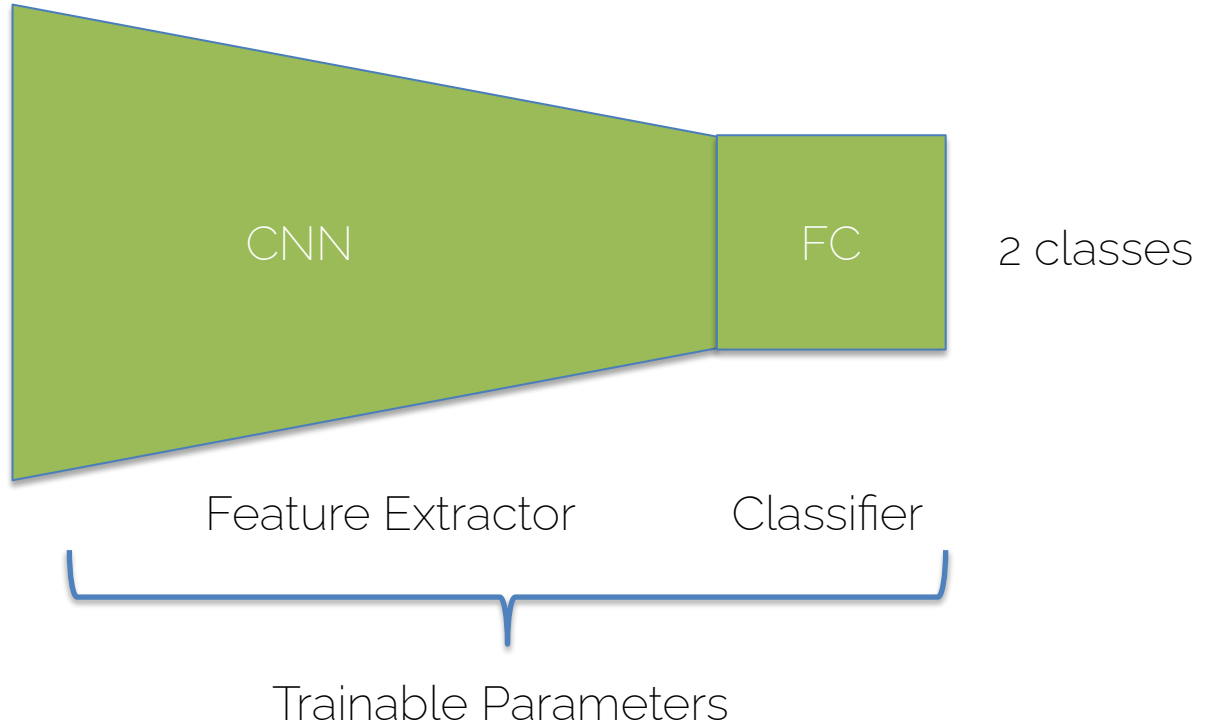  - Can we re-use (at least parts of) a pre-trained network for the new task?
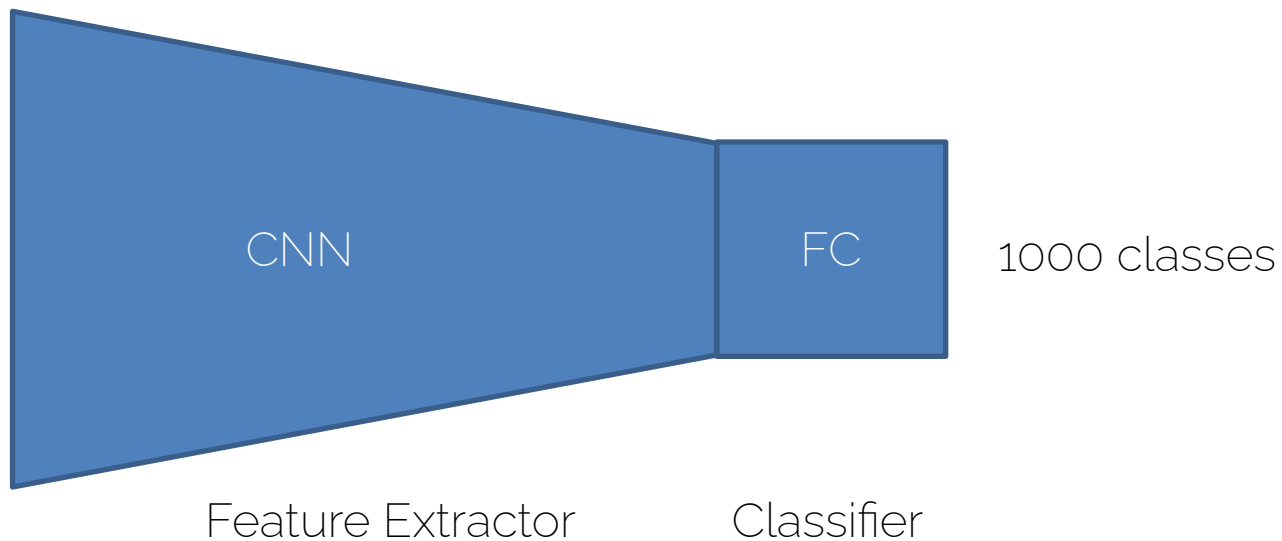
# Transfer Learning

Distribution
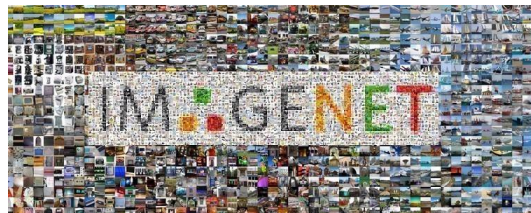
Distribution

P1

P2

Large dataset

Small dataset

Use what has been learned for another setting

# Transfer Learning



CNN

FC

2 classes

Feature Extractor      Classifier

Trainable Parameters

Coloring Legend:

Untrained

Trained

# Transfer Learning



CNN — FC — 1000 classes

Feature Extractor — Classifier

Coloring Legend:

- Untrained
- Trained

# Transfer Learning



CNN — FC — 2 classes

Feature Extractor | Classifier

Trainable Parameters

Coloring Legend:

Untrained

Trained

# Transfer Learning



CNN     FC     2 classes

Feature Extractor     Classifier

Trainable Parameters

Coloring Legend:

Untrained

Trained

# Transfer Learning



2 classes

Coloring Legend:

Untrained

Trained

Feature Extractor

Classifier

Maybe freeze weights/
slower learning rate/
nothing special

Newly initialized
head

# Application: Autoencoder

# 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



Input layer     Hidden layer     Output layer

"bottleneck"

Encoder     Decoder
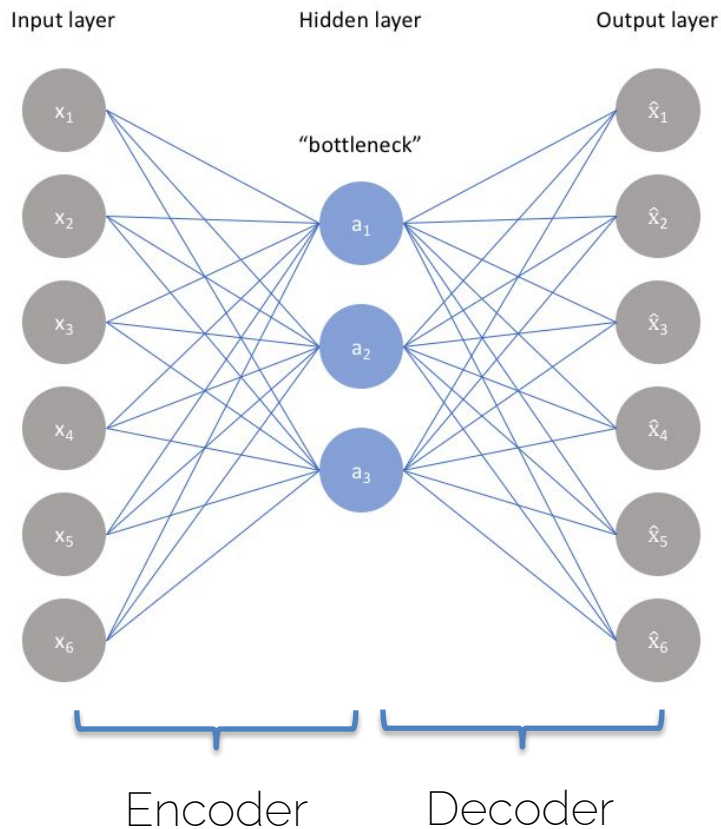
# Transfer Using an Autoencoder

- ## Step 1:
  - Train an Autoencoder on a large (maybe unlabelled) 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
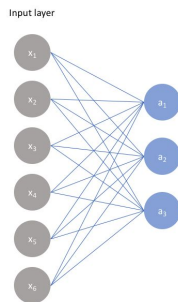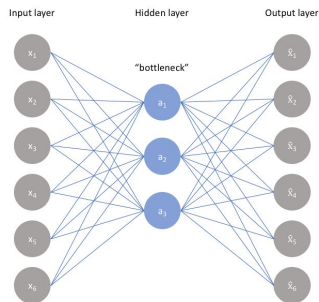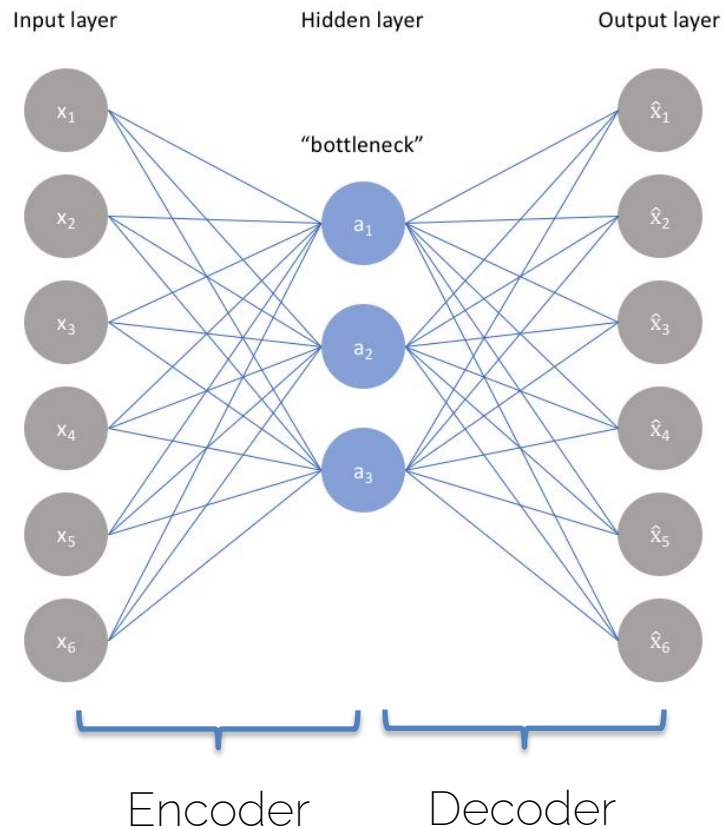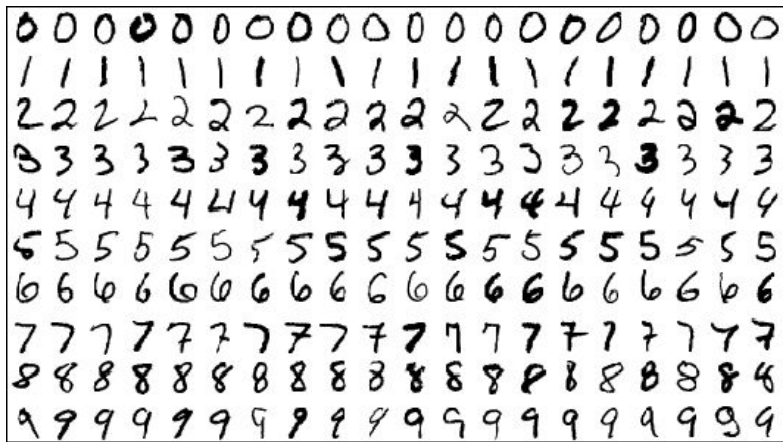
# Exercise 8

# Autoencoder

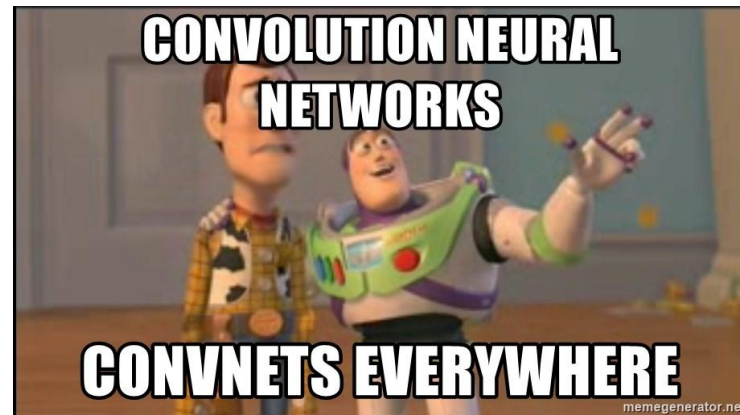- Exercise Task:
  - 60 000 Images
  - Only 300 with labels

MNIST database





Input layer    Hidden layer    Output layer

"bottleneck"

Encoder    Decoder

# We get there...

No convolutions yet,
but be prepared...

Next week will be the week.



But that means for now, we stick (one last time) with our
linear layers.

# Summary

- **Monday 30.06.2025:** Lecture 9
  - Convolutional Neural Networks 1
- **Wednesday 02.07.2025** 23:59:59 Exercise 8 Deadline
  - Autoencoder
- **Thursday 03.07.2025**: Exercise 9 Release
- **Monday 07.07.2025**: Lecture 10

# Good Luck
# &
# See you next time!