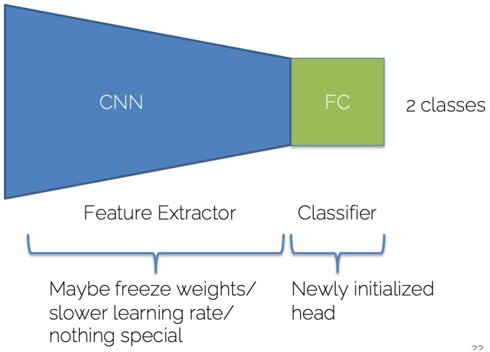
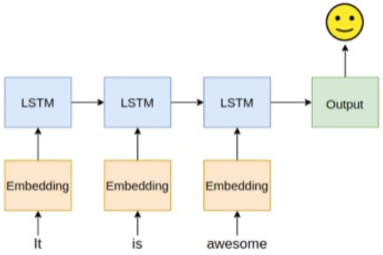
Introduction to Deep Learning (I2DL)

Exercise 11: RNNs and Q&A

Today's Outline

- Exercise 10 Review
- Recurrent Neural Networks
  - Exercise 11
- Live Q&A
  - and google form questions





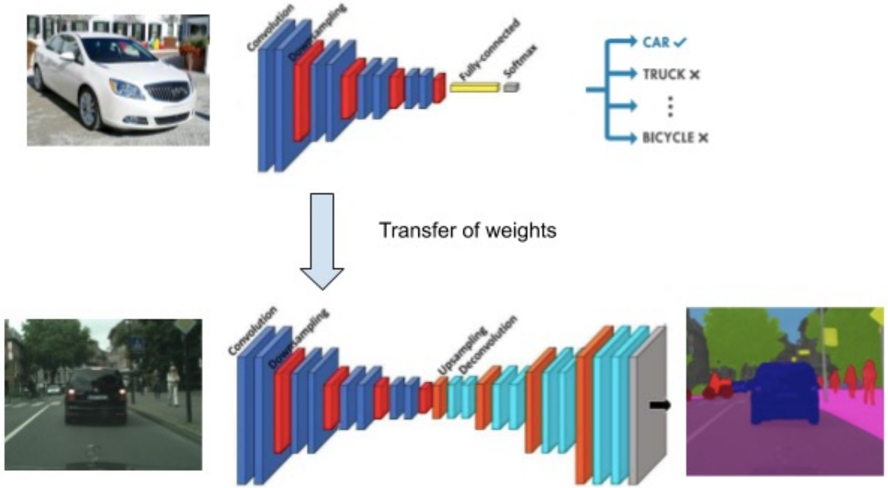
Exercise 10

Exercise 10: Semantic Segmentation

- **Goal:** Assign a label to each pixel of the image
- **Output of the network:** Segmentation mask with same shape as input image
- **Dataset:** MSRC v2 dataset, 23 object classes, contains 591 images with "accurate" pixel-wise labeled images

Suggested Approach

- **Idea:** Encoder-Decoder Architecture
- **Transfer Learning:** CNNs trained for image classification contain meaningful information that can be used for segmentation -> Encoder
- **Check out:** pre-trained networks like MobileNets



Leaderboard

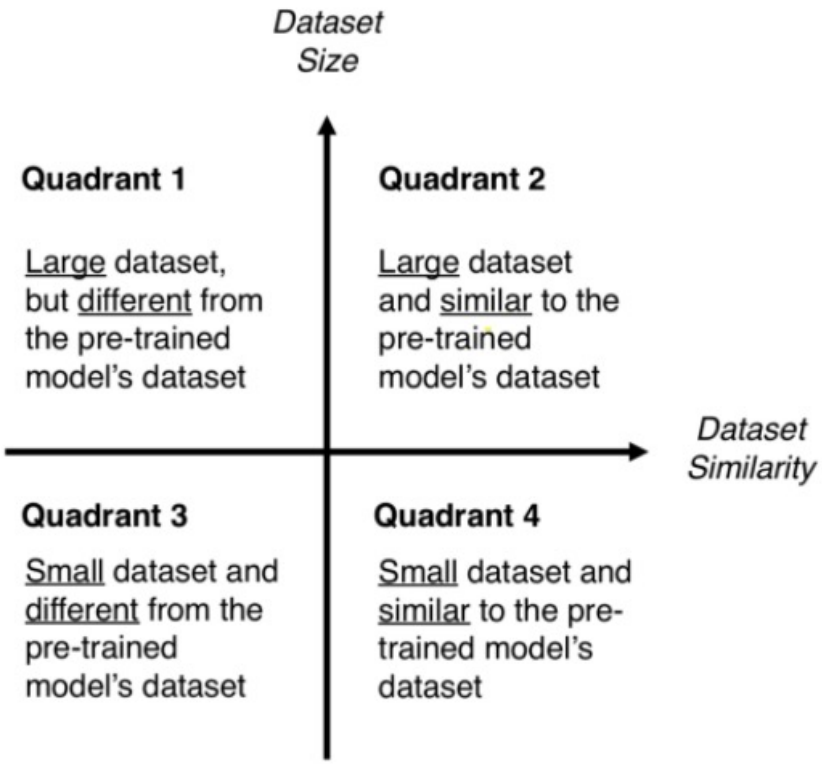
#	User	Score
1	u0575	93.15
2	u1219	92.88
3	u0088	92.38
4	u1191	92.11
5	u0943	91.62
6	u1240	90.93
7	u0169	90.89
8	u0158	90.86
9	u0798	90.64
10	u1540	90.27

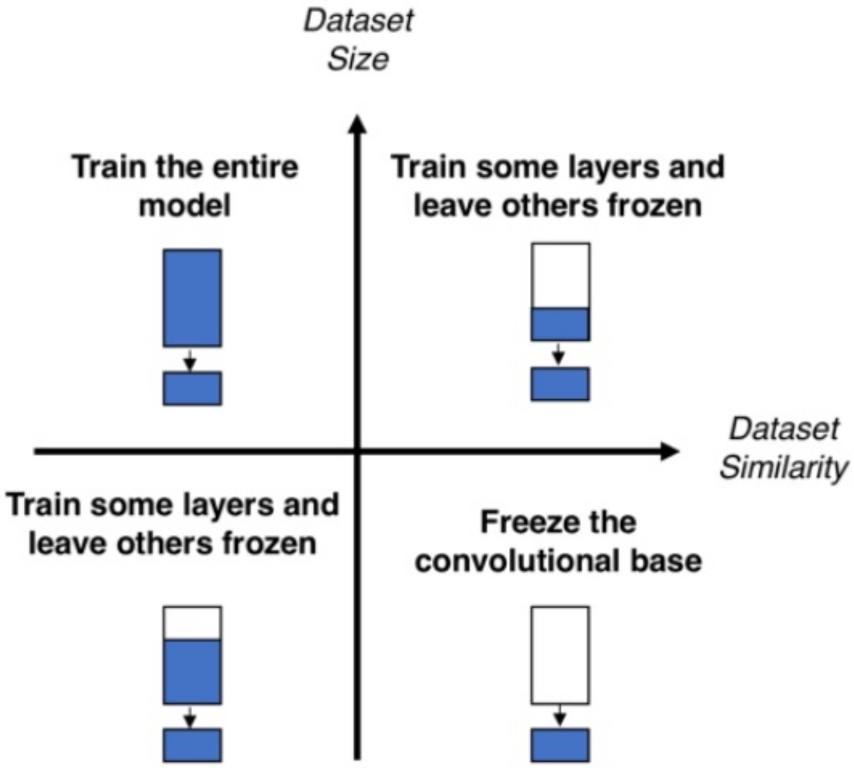
“Default” Approach (93.15)

- Take an already pretrained segmentation network
- Change the output layer to our number of classes
- Success!

```
from torchvision.models.segmentation import lraspp_mobilenet_v3_large
from torchvision.models.segmentation.lraspp import LRASPPHead

self.mobilenet = lraspp_mobilenet_v3_large(pretrained=True)
self.mobilenet.classifier = LRASPPHead(40, 960, hparams['n_classes'],
128)
```

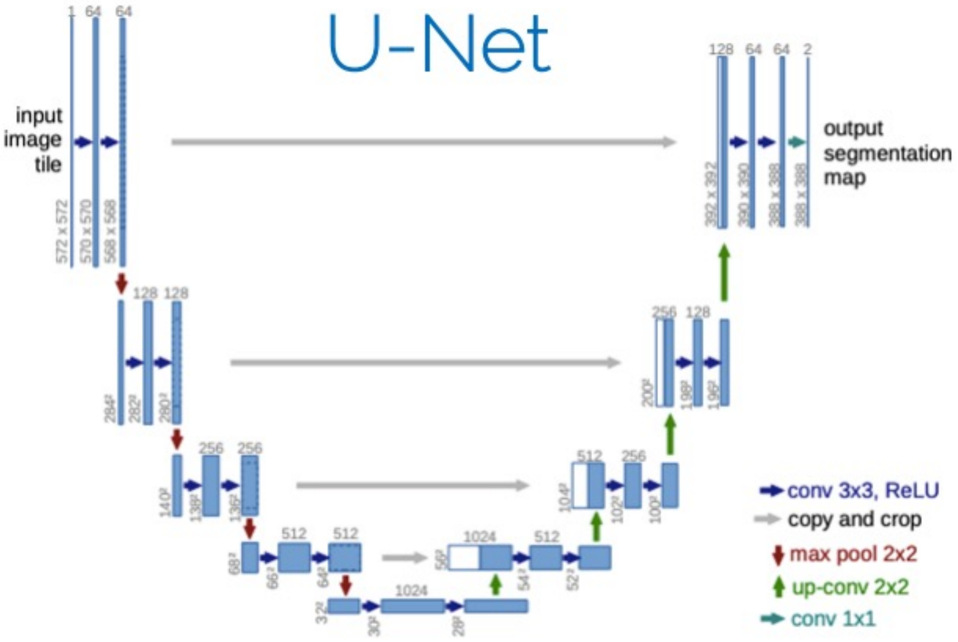
When/what to finetune?





So... something else? (92.38)

- Idea: Let's build a UNet



Let's start with pretrained backbone

- Get pretrained network and identify skip connection candidates

```
# get pretrained net
self.feature_extractor = mobilenet_v2(True).features
for params in self.feature_extractor.parameters():
    params.detach()
#output size should be: [-1, 1280, 8, 8]

# define forward hooks
# interesting layers:1:(16,120) 3:(24,60): 6:(32,30); 10:(64,15); 13:
(96,15); 16:(160,8); 17:(320,8); 18(1280,8)
self.horizontalLayerIndices = [6, 13, 18]
```

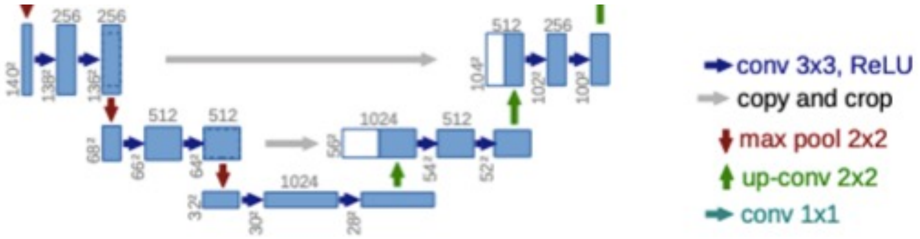
Let's check forward

- Forward backbone but keep track of skips
- "Bottleneck"
- Upsampling and filling in skips

```python
layeroutputs = []
for i in range(len(self.feature_extractor)):
    x = self.feature_extractor[i](x)
    if i in self.horizontalLayerIndices:
        layeroutputs.append(x)

x = layeroutputs[-1]
x = self.initialConv(x)

k = 3 if self.use30features else 4

for i in range(len(self.upsampler)):
    x = self.upsampler[i](x)
    if i < len(self.upsampler)-k:
        x = torch.concat((x, layeroutputs[-2-i]), dim=1)
    x = self.convs[i](x)
```

Some comments

- Bottleneck seems to be too tight
  - Make room for multiple non-linearities before upsampling and merging



- Bottleneck is using 8x8 with 1280 features
  - Use 1x1 convolutions to shrink down size first (we are not imagenet with 1000 classes where would need it)

```
0 | feature_extractor    | Sequential       | 2.2 M
1 | input_normalization  | Normalize        | 0
2 | upsampler            | ModuleList       | 0
3 | convs                | ModuleList       | 1.6 M
4 | initialConv          | Sequential       | 410 K
5 | lossFcn              | CrossEntropyLoss | 0
```

Some comments

- Good: usage of variables for filter/network size
  - Just don't hardcore numbers in your init unless you really want to keep them

```python
featureSize = np.linspace(featureSize, num_classes, 5)
featureSize = featureSize.astype('int')
```

- Don't forget to use data augmentation even when transfering weights
  - Could have been done outside of notebook, just a reminder ☺

Sample Outputs

Recurrent Neural Networks

Recurrent Neural Networks

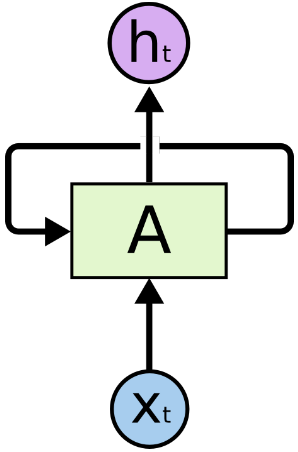
- **Idea:** Network that can capture the relationship between the inputs
- **RNNs:** Learning process is not independent
    - Remember things from processing trainings data
    - Remember things learnt from prior inputs, prior inputs influences decision
- **In other words:** RNNs produce different outputs for same input depending on previous outputs in the series.

$$\boldsymbol{A_t = \theta_c A_{t-1} + \theta_x x_t}$$

Previous hidden state

input

Output / Hidden State



Input

RNN Concepts

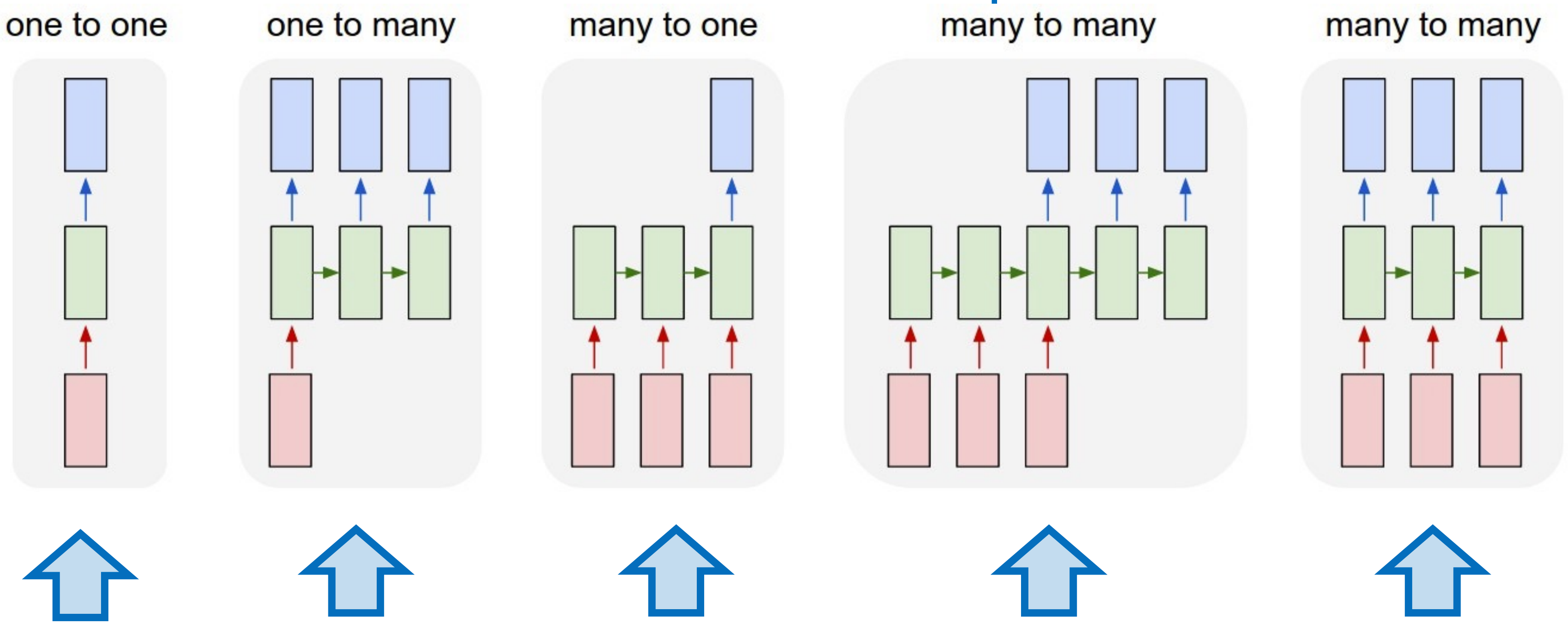


Image Classification

Image Captioning (image -> seq of words)

Sentiment Analysis (seq of words -> sentiment)

Machine Translation (seq of words -> seq of words)

Video Classification on frame level (seq of frames -> seq of class.)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Exercise 11

Exercise 11: Goal

Review: I wouldn't rent this one even on dollar rental night.

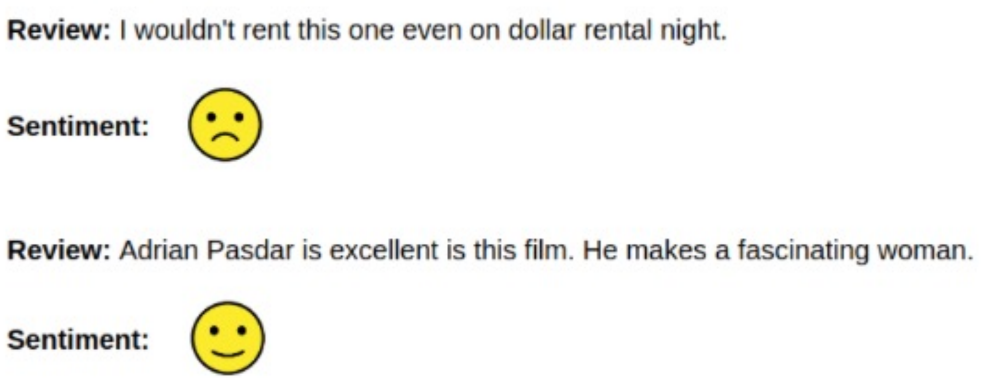
Sentiment: 

Review: Adrian Pasdar is excellent is this film. He makes a fascinating woman.

Sentiment: 

Exercise 11: Content

- Optional Notebook: RNNs and LSTMs
- Notebook 1: Text Preprocessing and Embedding
- Notebook 2: Sentiment Analysis

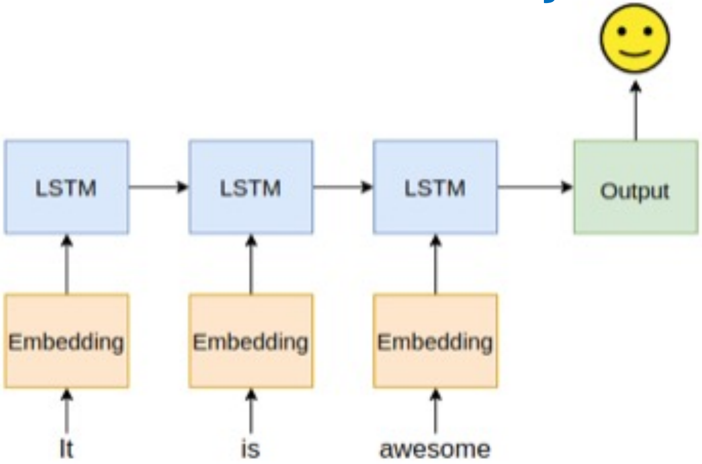


Notebook 1: Text Preprocessing and Embedding

- Sequential Data: from image data to text data
- Dataset: IMDb sentiment analysis dataset
- Goal of the notebook:
  - Data preparation
  - Implementation of Embedding layer

Notebook 2: Sentiment Analysis

- Network Architecture:
  - Embedding layer
  - RNN
  - Output layer,
    e.g. fully-connected layer
- Loss: Cross-Entropy Loss
- Performance measure: Accuracy
- Goal of the notebook: Implement and train a recurrent neural network for sentiment analysis



Submitted Questions

Organizational Questions: Exam

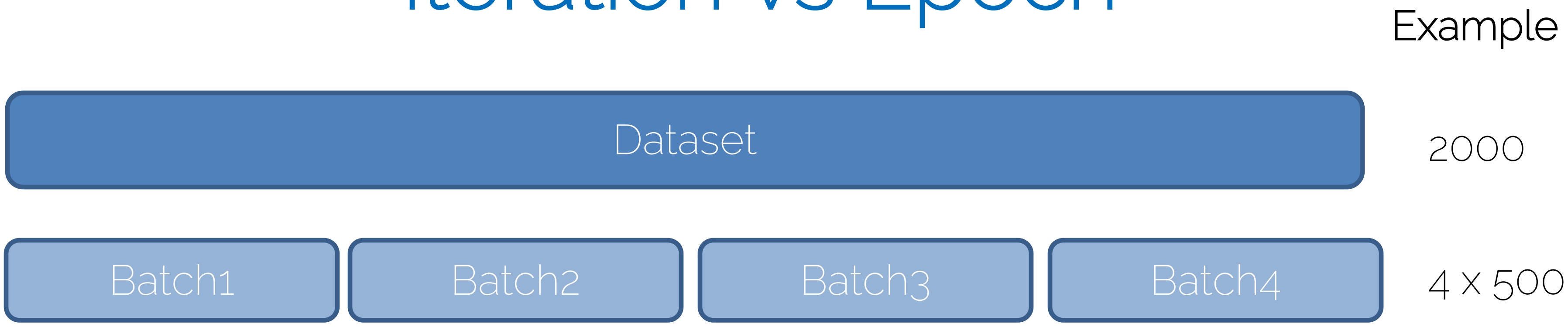
We will post more details about the exam procedure before the exam.

For all remaining questions, please use campuswire!

Content of Exam

- All Lecture Conten: Lecture 1-12
- All Exercise Content:
  - Tutorial Session Content
  - Notebooks

Iteration vs Epoch



Epoch
When an entire dataset is passed forward and backward through the network ONCE.

Batch
Subdivide dataset into smaller batches.
Batch Size = Number of training samples in one batch
Number of batches = Number of total batches to divide up entire dataset.

Iteration
Number of batches to complete one epoch.

Regularization

- Any strategy that aims to

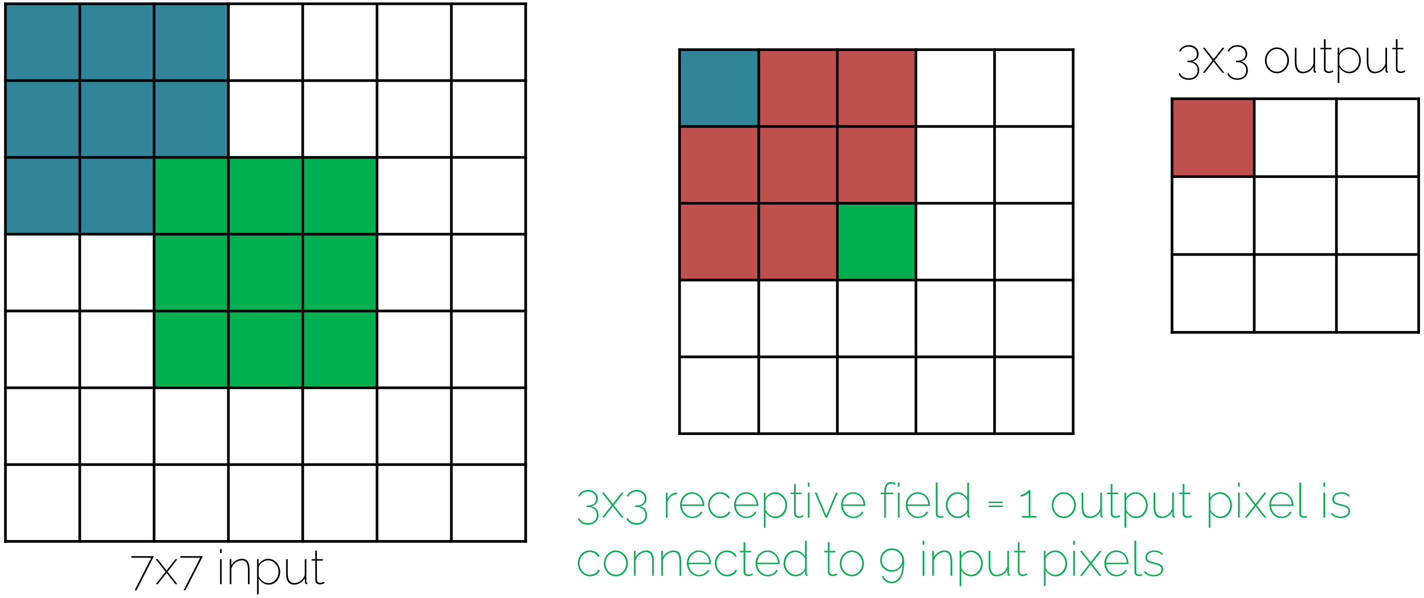
Lower validation error

Increasing training error

Regularization should make:

- Training harder
- Decrease generalization gap (difference between train/val)

Receptive Field



3x3 receptive field = 1 output pixel is connected to 9 input pixels

Q: How to calculate the receptive field of a Pixel after L layers using a general formular?

A:

Number of Channels in Convolution

Q: Why do I need to calculate the variables "in_channels"/"out_channels" in a CNN in Pytorch by myself? Since there is a formula behind this (out = ((n-f+2p)/s) +1), why can't the network in Pytorch do that on its own?

A: These are hyperparameters and depending on YOUR design choice.

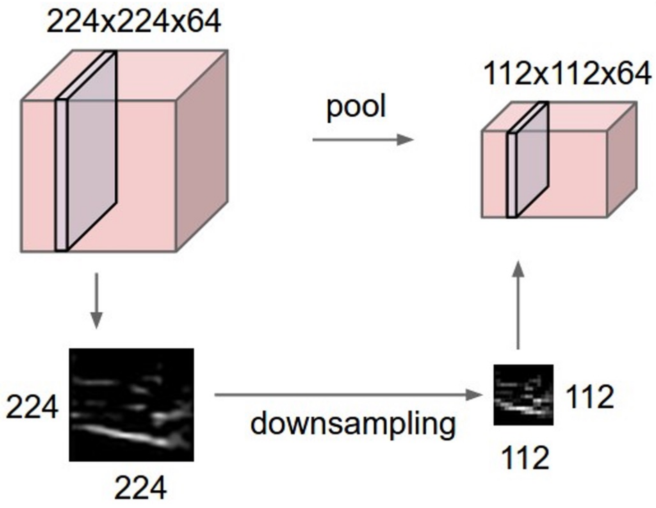
```
CLASS torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0,
      dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)
```

- Input: $(N, \boxed{C_{in}}, H_{in}, W_{in})$ or (C_{in}, H_{in}, W_{in})
- Output: $(N, \boxed{C_{out}}, H_{out}, W_{out})$ or $(C_{out}, H_{out}, W_{out})$, where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

Pooling Layer



- Conv Layer = 'Feature Extraction'
  - Computes a feature in a given region
- Pooling Layer = 'Feature Selection'
  - Picks the strongest activation in a region

Q: Isn't pooling applied individually for each channel (i.e. for each feature according to my understanding), how does it select features within a single channel?

A: Each filter is "responsible" for a specific feature, slides over input. The information about this feature contained in feature map. Pooling means selecting strongest features within a region.

Weight decay

Q: Why is L2 known as weight decay?

A: L2 regularization is considered equivalent to weight decay in case we use SGD. Not true, if we use other optimizers.

An Explanation:
https://towardsdatascience.com/weight-decay-l2-regularization-90a9e17713cd.

Adam : Bias Corrected

Q: Why is the biad correction in the ADAM optimizer?

- Combines Momentum and RMSProp

$$\boldsymbol{m}^{k+1} = \beta_1 \cdot \boldsymbol{m}^k + (1 - \beta_1)\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k)$$

$$\boldsymbol{v}^{k+1} = \beta_2 \cdot \boldsymbol{v}^k + (1 - \beta_2)[\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k) \circ \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k)$$

- $\boldsymbol{m}^k$ and $\boldsymbol{v}^k$ are initialized with zero
  $\rightarrow$ bias towards zero
  $\rightarrow$ Need bias-corrected moment updates

Update rule of Adam

$$\widehat{\boldsymbol{m}}^{k+1} = \frac{\boldsymbol{m}^{k+1}}{1 - {\beta_1}^{k+1}} \qquad \widehat{\boldsymbol{v}}^{k+1} = \frac{\boldsymbol{v}^{k+1}}{1 - {\beta_2}^{k+1}} \longrightarrow \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \cdot \frac{\widehat{\boldsymbol{m}}^{k+1}}{\sqrt{\widehat{\boldsymbol{v}}^{k+1}} + \epsilon}$$

In-/Equivariance

Have: function f:A->B,
group action g on A, group action g' on B

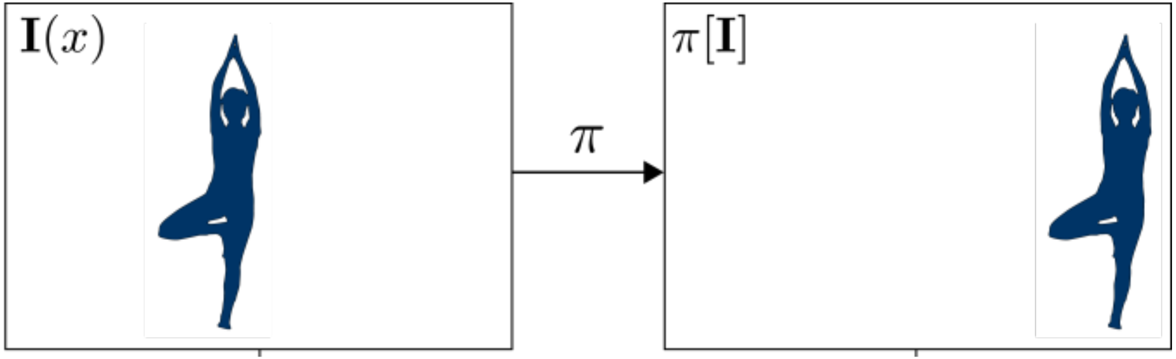
- Invariance

$$f(g(I)) = f(I)$$

- Equivariance

$$f(g(I)) = g'(f(I))$$

Convolutions are translation equivariant



More Information

- Q: What about rotation?
  A: That's why we need data augmentation.

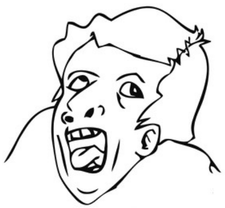
- Q: But my friend said convolutions are translation invariant!
  A: They are not. But a classification network could be. Consider the ResNet structure:
  - Convolution backbone
  - Global Average Pooling (GAP) layer
  - Fully Connected layers

Batchnormalization before/after Relu?

```python
layer = nn.Sequential([
        nn.Conv2d(inp, out),
        nn.BatchNorm2d(out),
        nn.RelU()
])
```

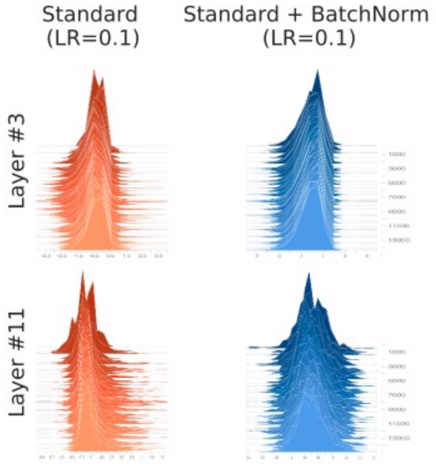
```python
layer = nn.Sequential([
        nn.Conv2d(inp, out),
        nn.RelU(),
        nn.BatchNorm2d(out)
])
```

- A: It's a hyperparameter
  - Original paper: before
  - Many others (including myself): after



Batchnormalization before/after Relu?

- Why it might be not important where to put it https://arxiv.org/pdf/1805.11604.pdf
  - Gist: gradient landscape smoothing is the biggest contributor which it does in both positions



- Practical use-cases:
  - After: More efficient on imagenet: https://github.com/ducha-aiki/caffenet-benchmark/blob/master/batchnorm.md
  - Before: computationally more efficient

Does Batchnormalization introduce a non-linearity?

- Formula

$$\hat{x} = \frac{x - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 - \epsilon}}$$

- Therefore:
  - During train: mean/variance depend on x -> non-linear
  - During test: fixed mean/variance

Random network questions

- Q: When it's said "increase the number of parameters"/"capacity" in a NN. What does this exactly mean?

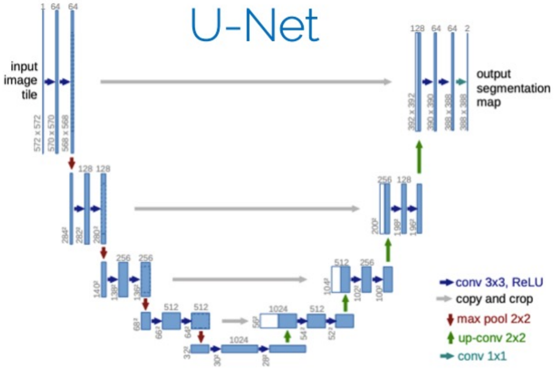
```
  | Name                 | Type             | Params
---------------------------------------------------------
0 | feature_extractor    | Sequential       | 2.2 M
1 | input_normalization  | Normalize        | 0
2 | upsampler            | ModuleList       | 0
3 | convs                | ModuleList       | 1.6 M
4 | initialConv          | Sequential       | 410 K
5 | lossFcn              | CrossEntropyLoss | 0
---------------------------------------------------------
4.2 M     Trainable params
0         Non-trainable params
4.2 M     Total params
16.956    Total estimated model params size (MB)
```

- Q: Do we need to know architectures by heart
  A: No, but layers like ResNet blocks

Unet and Skip Connections

- Q: Are skip connections solving the vanishing gradient problem in a Unet?

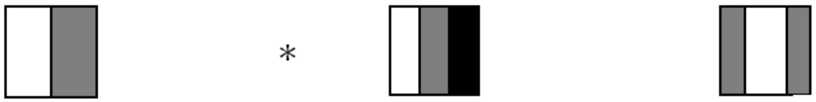
- A: Yes, but that's not the sole reason why we are using them



Conv Filter Application

When designing a convolutional filter, how do we know what values to use (-1, 0, or 1)? i.e. what does these values mean or what is the impact of them?
1 0 -1		-1 0 1
1 0 -1		-1 0 1
1 0 -1		-1 0 1
For example, what may be the difference between the two filters above?

$$\begin{array}{|c|c|c|c|c|c|}\hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline \end{array}$$



https://upscfever.com/upsc-fever/en/data/deeplearning4/2.html

Audience Questions

See you next week ☺