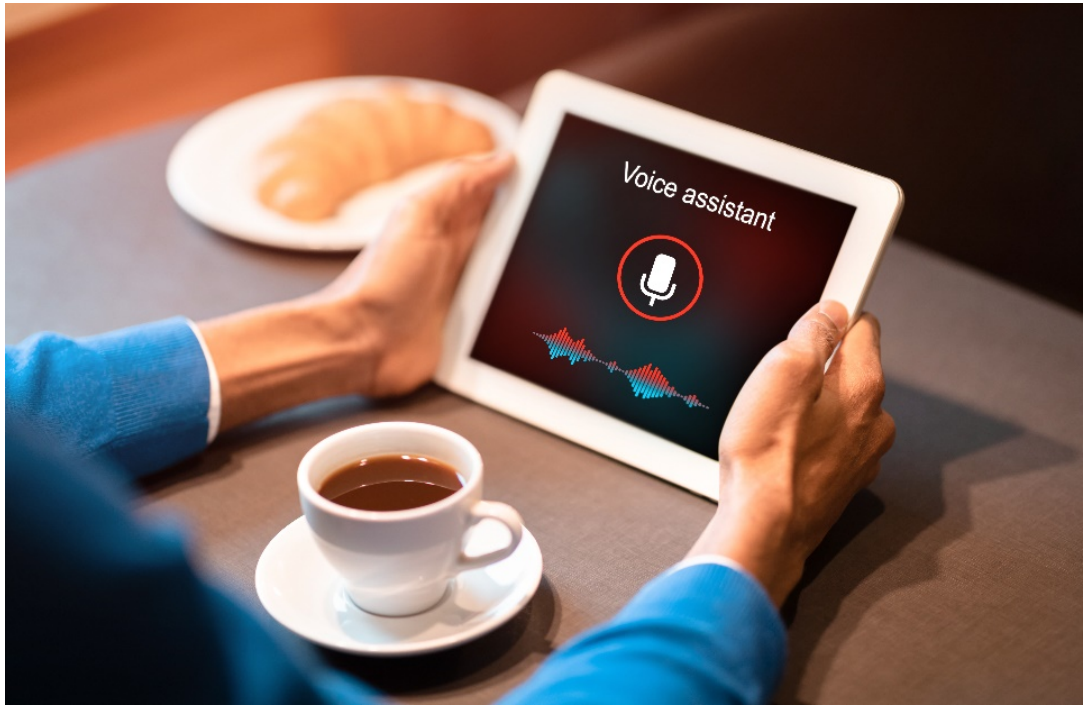
ML school audio track

Deep Learning II

Dr. Paul Wallbott



Organisation

Table 1

Date // Time	Mo	Tu	We	Do
9.30 – 11.00		Exercise I	Exercise III	
11.15 – 12.45		**Deep Learning I**	**Keyword Spotting**	
13.00 – 14.15				
14.15 – 15.45	**Welcome**	Exercise II	Exercise IV + **Wrap Up**	
16.00 - 17.30	**Intro to Audio**	**Deep Learning II**		

- Rough schedule, might change during the days
- Exercises in small groups
  - You will fill in the missing code in jupyter notebooks

Open Questions and exercise feedback

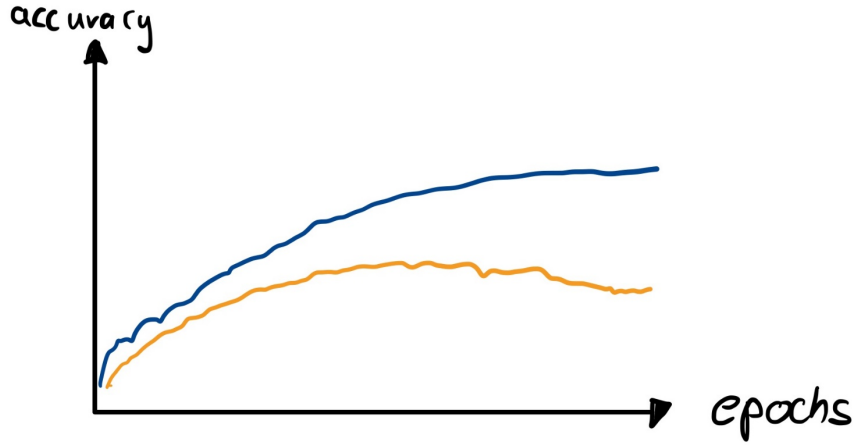
- Lets go over the notebook together

Agenda

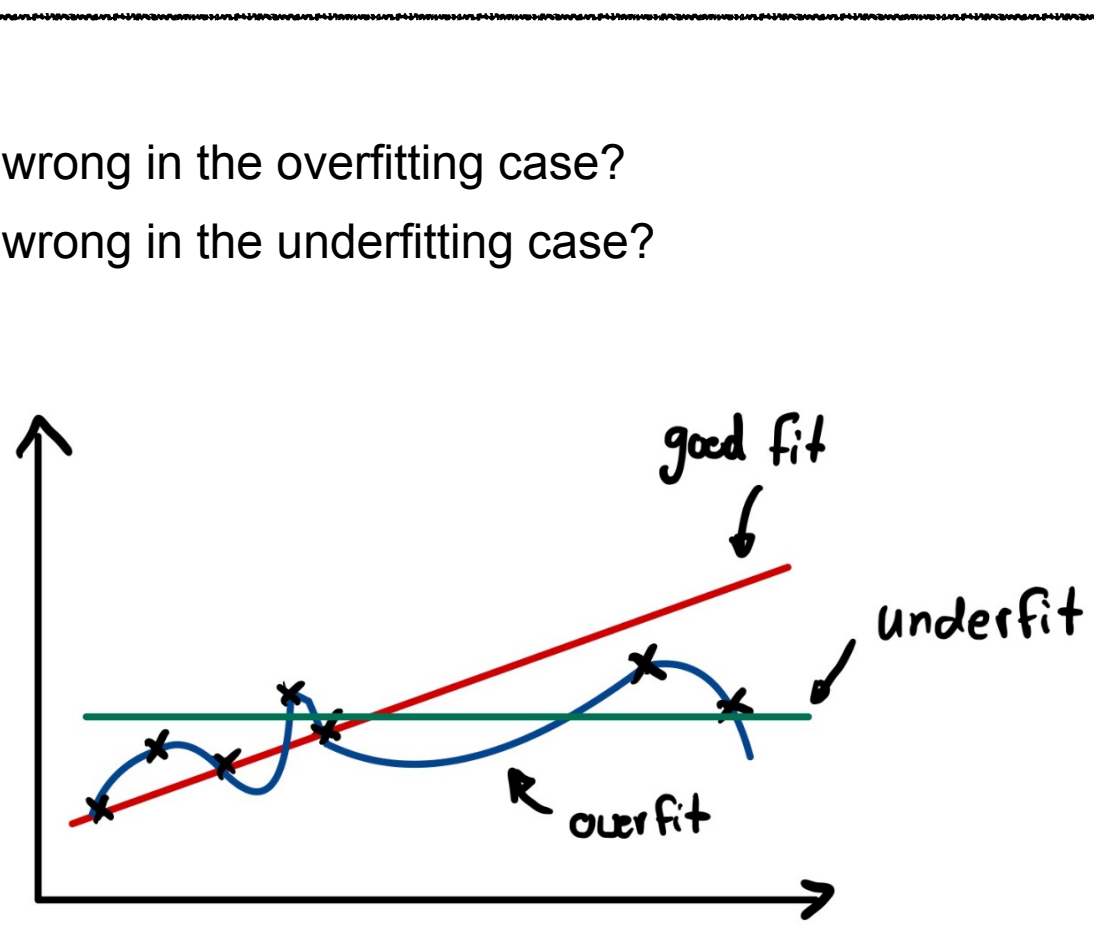
- We will discuss some typical problems when training DL models
  - Overfitting / underfitting
    - L1, L2 Regularization
    - Weight decay
    - Dropout
  - Data sparsity
    - Data augmentation
- Advanced model architectures
  - Convolutional neural networks
  - Temporal convolutions
  - Layer types (pooling, flattening, batch norm, …)

Overfitting diagnosis

- We have discovered overfitting
  - Train accuracy goes up while val accuracy stays constant or drops
  - Large gap between accuracies on val and train set



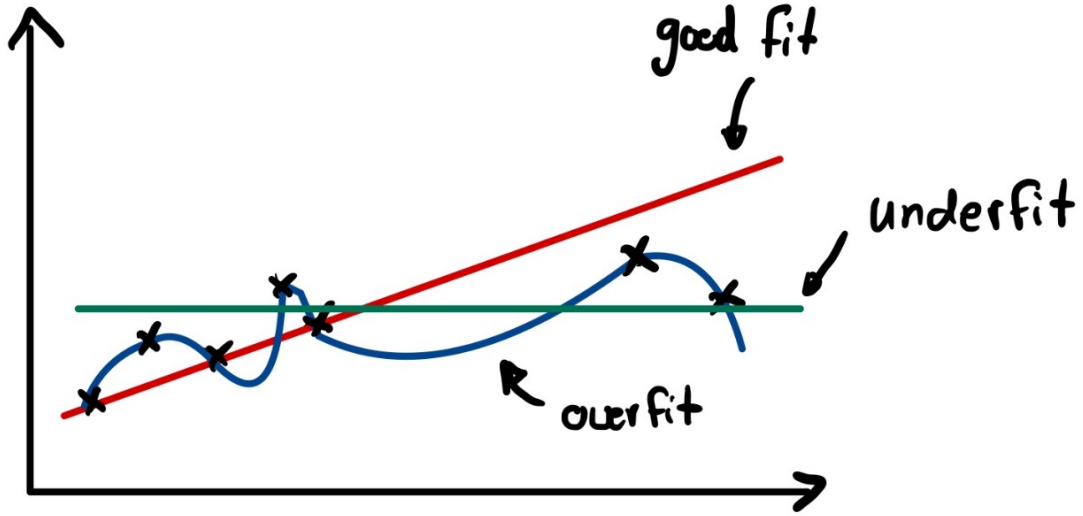
Mini exercise



- What is going wrong in the overfitting case?
- What is going wrong in the underfitting case?

Solution

- What is going wrong in the overfitting case?
  - Too many parameters
  - Outliers
- What is going wrong in the underfitting case?
  - Not enough parameters



Underfitting

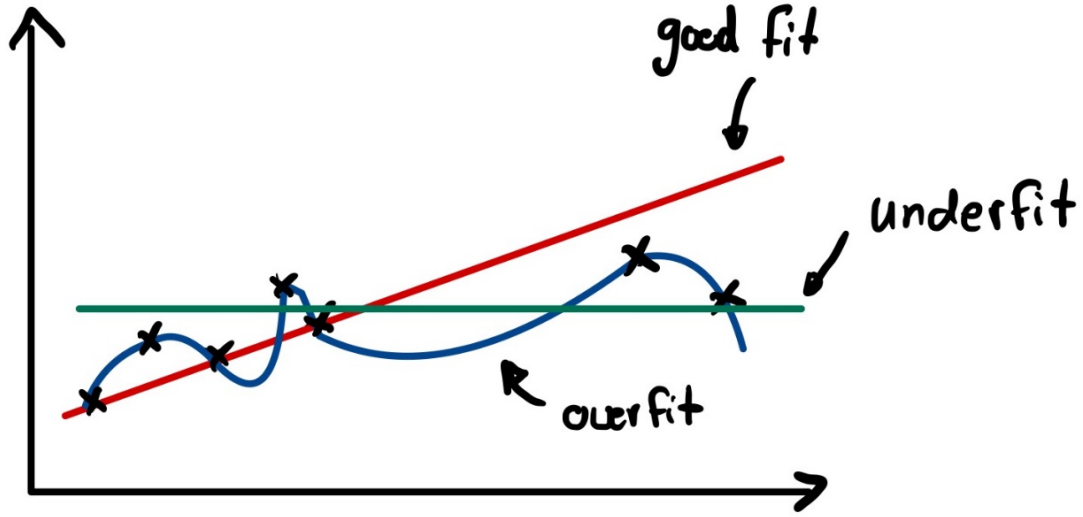
- Model is not powerful enough (e.g., higher order polynomial)
- features could be better
  - Data distribution can be explained with x,y
  - Alternatively use the right feature r --> one variable sufficient

$x^2+y^2 < R^2$ (x, y)

$r < R$ (r, ρ)

Overfitting

- Training process
  - Stop training early „early stopping“
- Manual degree of freedom reduction
  - less layers or neurons per layer
- Data pre processing
  - Remove outliers
- Regularization
  - Methods that reduce degree of freedom
  - Physics: Enforce symmetries!



Regularization I: L1, L2

- Weight penalty term in loss
  - unnecessary weights driven to 0
  - Reduces the number of parameters of the final model
- Most common types
  - L2 regularization / weight decay
  - L1 regularization (used less often)
- Keras adds regularizer layerwise
  - “kernel_regularizer = keras.regularizers.l2(0.01)”
  - Allows to add only some weights to the loss

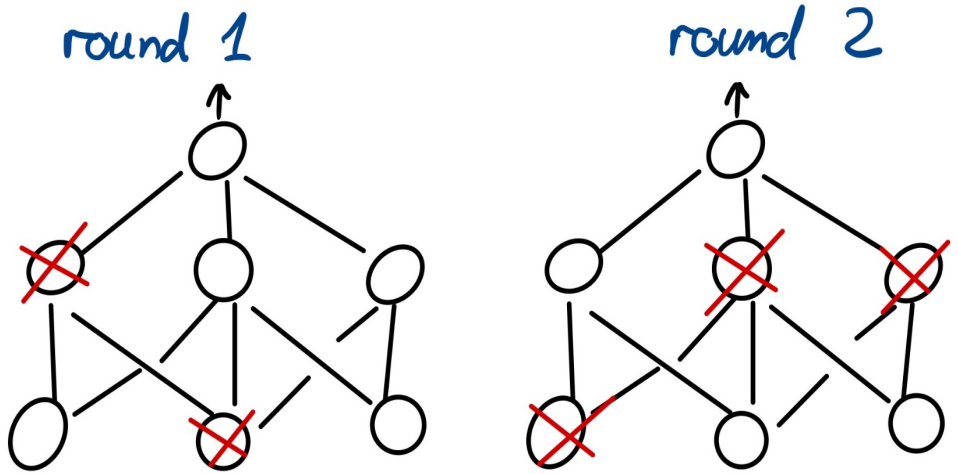
$$L \rightarrow L + \underbrace{\sum_{i=1}^{k} w_i^2}_{L_2}$$

$$L \rightarrow L + \underbrace{\sum_{i=1}^{k} |w_i|}_{L_1}$$

Regularization II: Dropout

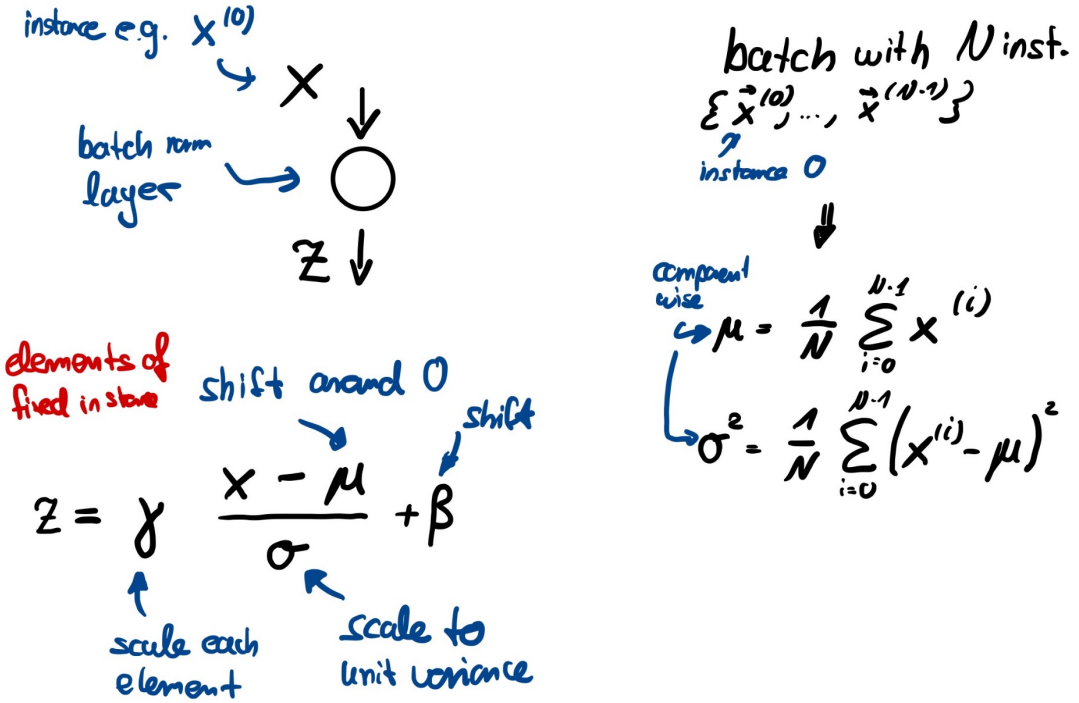
Hinton et al., 20212

- Idea:
  - All neurons can randomly drop out with probability p of a training step
  - A group of distinct but similar models are trained (ensemble), that share weights
  - On inference (test, validation) p = 0
- Practical application
  - Training choose p in [0.1,0.5]
  - Keras has extra dropout layer: keras.layers.Dropout
- Improves almost all trainings



Regularization III: Batchnorm

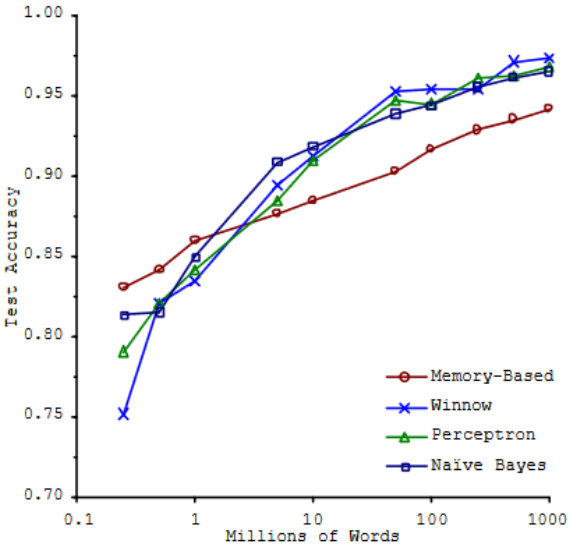
- Mean 0 and variance 1 normalization improves generalization of networks for several reasons
- Want this at several points in the network!
- But what are mean and variance of the dataset at particular point in neural network?
  - Use all data in batch for component wise optimization
- Keep moving average of mu and sigma in memory during training
  - During **inference**, use whole training sets average instead of batch for statistics



Frei nach https://en.wikipedia.org/wiki/Batch_normalization

Regularization IV: Data augmentation

- More data is better!
  - Language ambiguation task
  - Can prevent overfitting (previous slides)
- More data are expensive --> augmentation
  - Use data we have to create more
- How to create more data?
  - Vary data quality (e.g., resolution)
  - Vary experimental conditions (e.g, lighting)
  - Force network to learn symmetries (rotation, shifts)



https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/acl2001.pdf

Mini Exercise

- Create a list of augmentation measures applied directly to audio data
  - Specifically for the keyword spotter, what could be varied to obtain more robust results?
  - Could help to think about humans listening to others
- We train on image data (spectrograms). What could we use for images?

Regularization IV: Audio augmentation

- Audio
  - Pitch (how high or low the speaker speaks)
  - Speed
  - Amplitude (loud?)
  - Different **background noises** and noise levels
- Spectrograms
  - **Translation** to recognize parts of word (why not rotations?)
  - Covering random parts of the image (spec augment)

CNNs I: Definition + Intuition

- Convolutional neural networks (CNNs)
- Inspired by classical image processing
  - Grayscale image is matrix of values (Time x Frequency)
  - Apply a filter to generate a new image (e.g., that shows the edges)
  - Mathematically apply kernel (often 3x3) with fixed weights (9)

-1	0	1
-1	0	1
-1	0	1

custom

- Drag a fixed size kernel over the image to get a new image
- Perform weighted sum + activation

https://setosa.io/ev/image-kernels/

CNNs I: Definition + Intuition

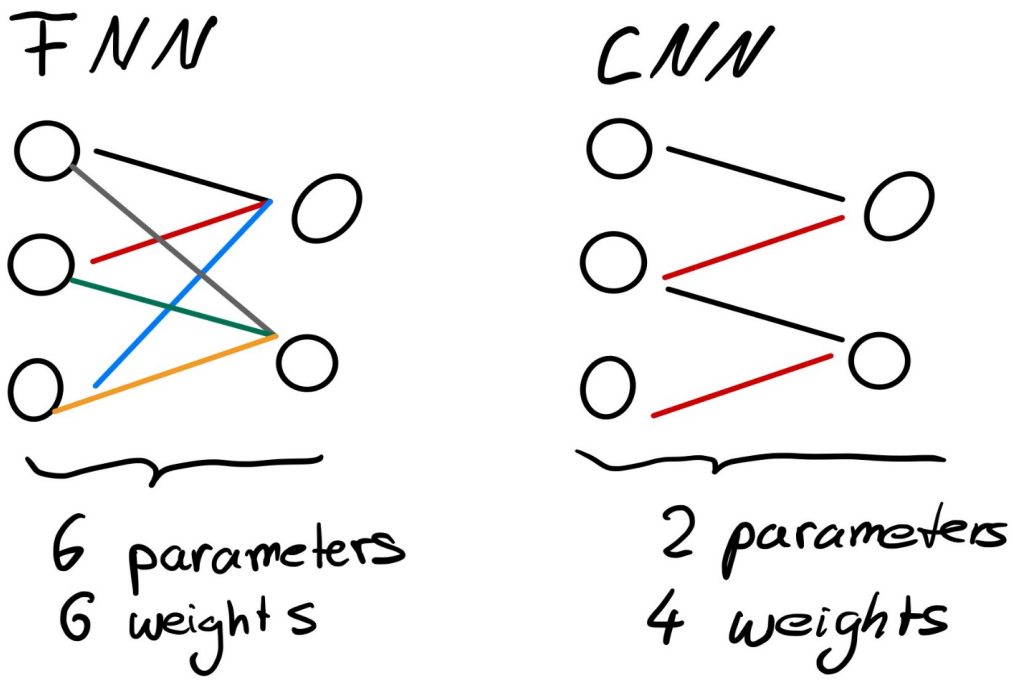
- Convolutional layer applies **n kernels**
- **Each kernel** has
  - A size / receptive field of e.g. 3x3 for 2D convolution, grayscale image
  - unique weights (w1,…,w9)
  - Produces 1 distinct feature map, aka new image
  - **Weights are learned** during training
- CNN layers have very few parameters and can learn great features
  - Example is 1D: advantage scales with dimensionality!

Mini Exercise

- What is the difference between a fully connected feed forwared network and a CNN? Explain it using the following example:
  - Draw a fully connected layer with 3 input and 2 output neurons
  - Draw a convolutional layer with 3 input neurons 2 output neurons and a kernel of size 2 (or 2 x 1) and one feature map
  - How many unique weights do the 2 variants have?

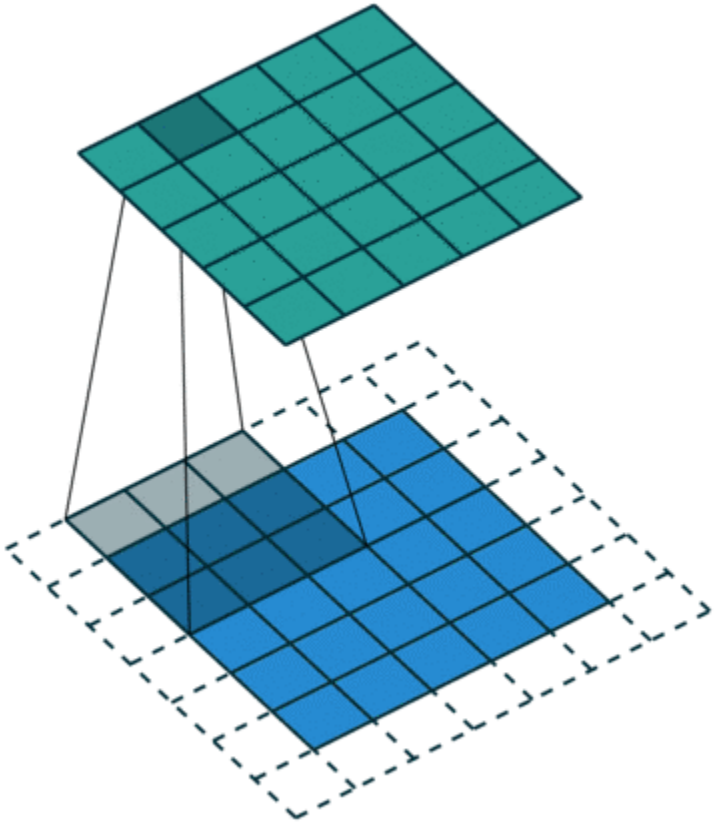
Solution

- What is the difference between a fully connected feed forwared network and a CNN? Explain it using the following example:
  - Draw a fully connected layer with 3 input and 2 output neurons
  - Draw a convolutional layer with 3 input neurons 2 output neurons and a kernel of size 2 (or 2 x 1) and one feature map
  - How many unique weights do the 2 variants have?



CNNs II: Padding, Pooling, Stride

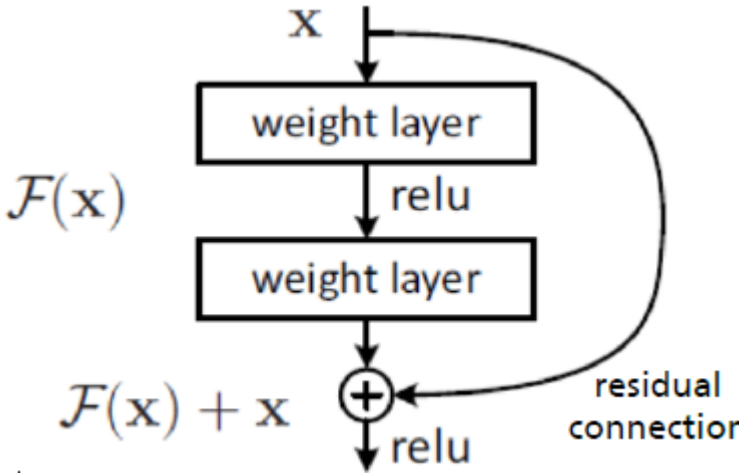
- Lets take the model builders perspective
- To get same size output feature map use **padding** (add frame of 0s to original image)
- To reduce feature map dimensionality use
  - **stride.** How far do we jump when sliding window over image?
  - **pooling layers** (with pooling size). Combine local groups of pixels to one pixel (average, max, min)
- Great visualization of all types of convolutions here



https://theano-pymc.readthedocs.io/en/latest/tutorial/conv_arithmetic.html

CNNs III: resnet

- Want deep architectures for good performance (many parameters)
- Problem: E.g., vanishing gradient over to many layers
- Solution: Use skip connections and learn **residuals** (resnet)

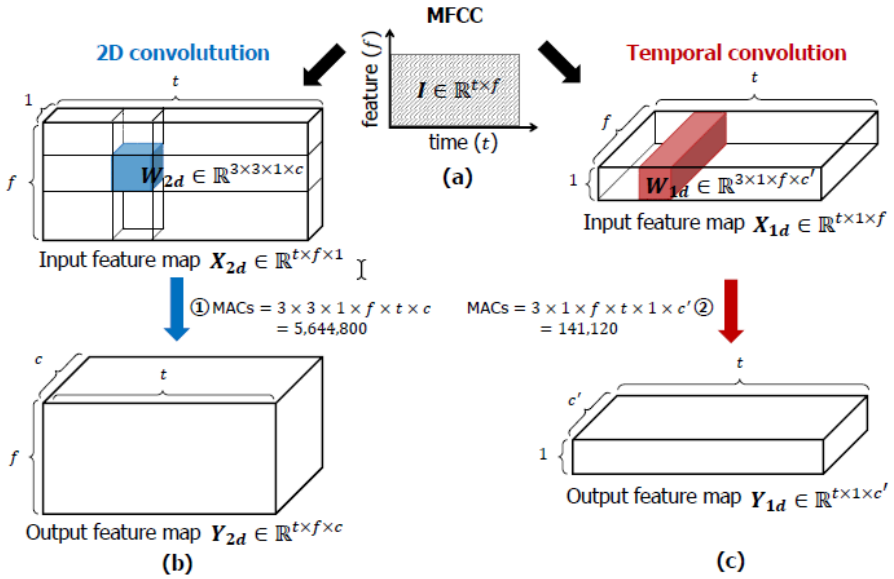


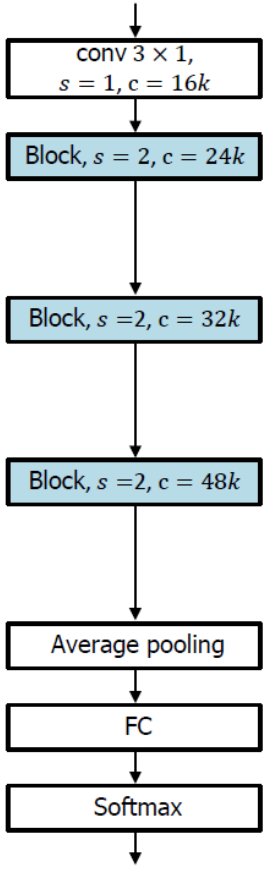
Source: He et. Al 2016: Deep residual learning

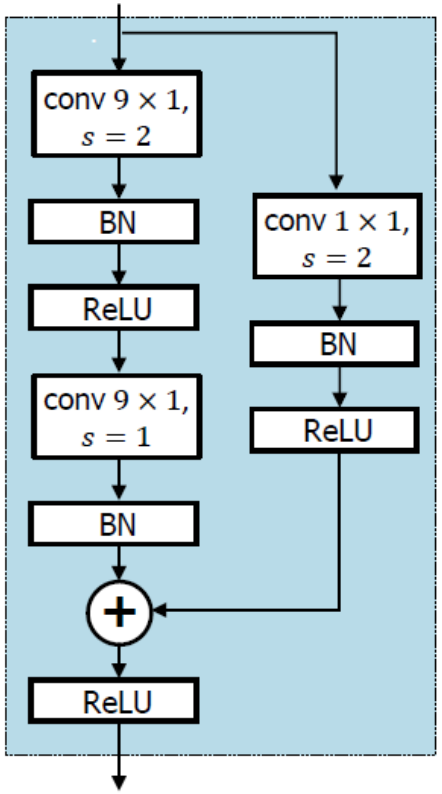
CNNs IV: Convolutions in time

- Could think of MFCCs as (T x F x 1) grayscale image
  - use 2D convolutions for keyword spotting
- Think of it as time series (T x 1 x F) or (T x 1) image with F color channels!
  - 1D convolution in time now considers all frequencies simultaneously
  - Decrease parameters while first layer already knows all frequency input
  - Smaller output image
- Can use many layers on the reduced size image

CNNs V: TC resnet







take all frequencies from 1 time step from the spectrogram

Source: Choi et al. , 2019

Exercise

- We will use what we have learned to improve our keyword spotter
- Work through exercise 3b
  - Add the new model
  - Add the data augmentation pipeline to create more data
  - Add dropout etc.
- Goal: In the end we should have a good classifier