Tutorial:

Deep Learning for Music Classification using Keras

Part II

Deep Learning for Music Classification using Keras

Alexander Schindler
Scientist
AIT & TUWien

Alexander.Schindler@ait.ac.at
http://ifs.tuwien.ac.at/~schindler

Thomas Lidy
Head of Machine Learning
Musimap

tom@musimap.com
www.musimap.com

Winners of 3 International Benchmarks

with Deep Learning on Audio

Thomas Lidy & Alex Schindler:

Winner IEEE DCASE 2016
Domestic Audio Tagging Contest

Winner MIREX 2015
Music/Speech Categorization

Thomas Lidy & Alex Schindler:

Winner MIREX 2016
Music Genre and Mood Classification

Tutorial Agenda

I. Deep Learning Basics:

- Audio Processing Basics
- History of Neural Networks
- What is Deep Learning
- Neural Network Concepts
- Coding Examples

II. Convolutional Neural Networks:

- Difference CNN – RNN
- How CNNs work (Layers, Filters, Pooling)
- Application Domains and how to use in Music
- Coding Examples

Tutorial Agenda

III. Instrumental, Genre and Mood Analysis:

- Large-scale Music AI at Musimap
- Instrumental vs. Vocal Detection
- Genre Recognition
- Mood Detection
- Coding Examples

IV. Advanced Deep Learning:

- Similarity Retrieval
- Siamese Networks
- Learning Audio Representation from Tag Similarity
- Coding Examples

Tutorial on Github

https://github.com/slychief/mlprague2018_tutorial

or

bit.ly/mlmusic18

+ scroll to the end of the page to download the data sets!

- GTZAN Music Speech
- MagnaTagaTune

Thomas Lidy – Short Bio

- 14 years of Machine Learning experience

- since 2004 Machine Learning on Audio

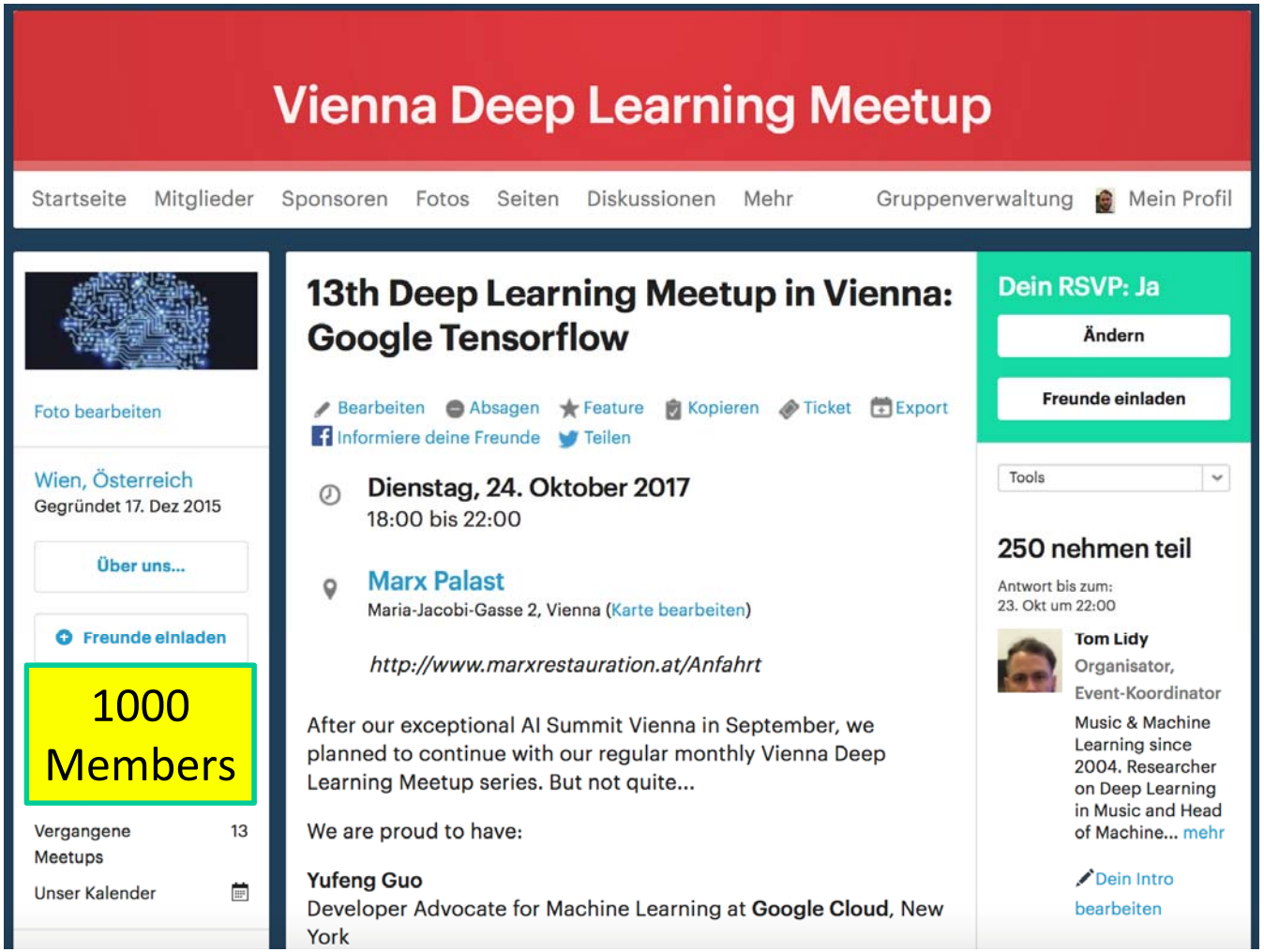
- 2004-2012 Researcher TU Wien – ML & Music
- 2008-2013 CEO Spectralmind – music discovery
- 2014-2015 Head of Product, myr:conn - data mining
- 2015-2016 Researcher TU Wien – DL, #MusicBricks
- **since 2016 Head of Machine Learning, Musimap**

TU WIEN TECHNISCHE UNIVERSITÄT WIEN

SPECTRAL MIND

musimap

Organizers and Hosts: Vienna Deep Learning Meetup



https://www.meetup.com/Vienna-Deep-Learning-Meetup

Credits / Contributions

Contributions & sources:

- Jan Schlüter (OFAI Vienna, www.ofai.at/~jan.schlueter)

- Yoshua Bengio
- Andrej Karpathy‘s blog
- and many others…

Questions to the audience

- Who has worked with Neural Networks before?
- Who currently uses Deep Learning?
- If so, which software do you use?
  - Caffe? Torch? Theano? Tensorflow? Keras? Lasagne?

    Others?

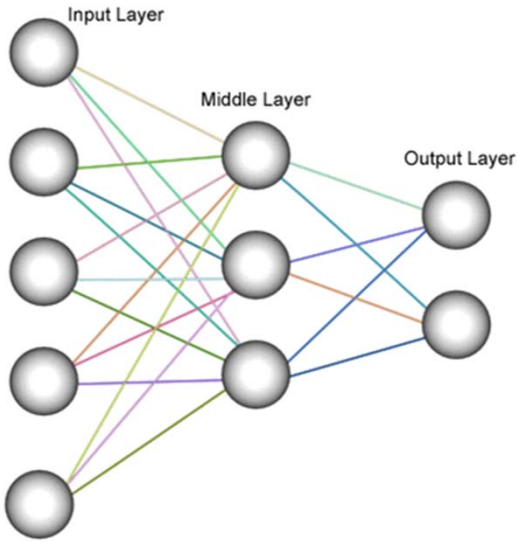
Deep Learning

= (Deep) Artificial Neural Networks (ANNs)

Neural Networks are loosely inspired by biological neurons that are interconnected and communicate with each other

Neural Networks

In reality, a neural network is a **mathematical function**:

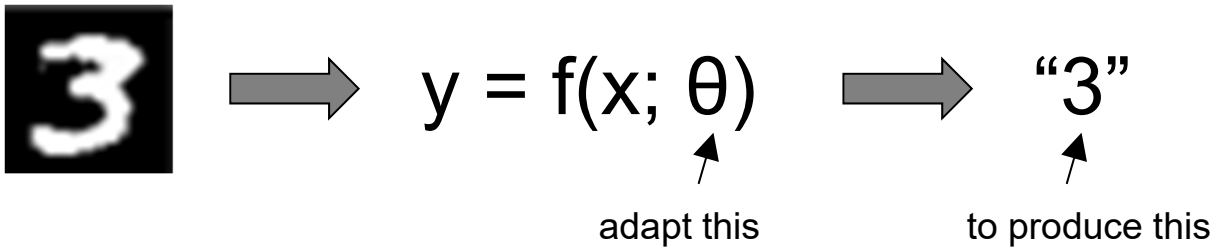


- in which the **“neurons”** are **sets of adaptive weights**, i.e. numerical parameters that are **tuned by a learning algorithm**
- which has the capability of approximating non-linear functions of their inputs

Src: Img: https://devcentral.f5.com/weblogs/images/devcentral_f5_com/weblogs/macvittie/WindowsLiveWriter/Applicationdeliveryneuralnetworksandinte_3132/ann_2.png
Text: https://en.wikipedia.org/wiki/Artificial_neural_network

What is Deep Learning?

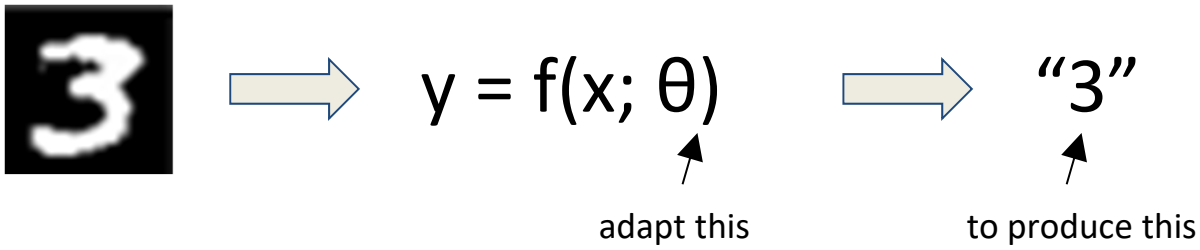
- **Machine learning:** Express problem as a function, automatically adapt parameters from data



- **Deep learning:** Learnable function is a **stack of many simpler functions** that often have the same form
  - Often, it is an artificial **neural network**
  - Often, one tries to minimize hard-coded steps

What can we do with Deep Learning?

Most research focuses on “narrow AI” that solves a single task.



Many tasks can be cast into such a function learning problem and solved with deep learning

But they are focused on solving an isolated task => no general Artificial Intelligence yet

Let's Get Concrete

- **Machine learning:** Express problem as a function, automatically adapt parameters from data
- **Deep learning:** Learnable function is a stack of *many simpler functions* that often have the same form

$$y = f(f(f(x;\theta_1);\theta_2);\theta_3)$$

"3"

adapt this

to produce this

- Most commonly used functions:
  - Matrix product: $f(x; \theta) = W^T x$
  - 2D Convolution: $f(x; \theta) = x * W$
  - Subsampling
  - Element-wise nonlinearities (sigmoid, tanh, rectifier)

Mathematical Reasons for Going “Deep”

A neural network with a **single hidden layer** of enough units can approximate **any continuous function** arbitrarily well. In other words, it can solve whatever problem you’re interested in! (Cybenko 1998, Hornik 1991)

Mathematical Reasons for Going "Deep"

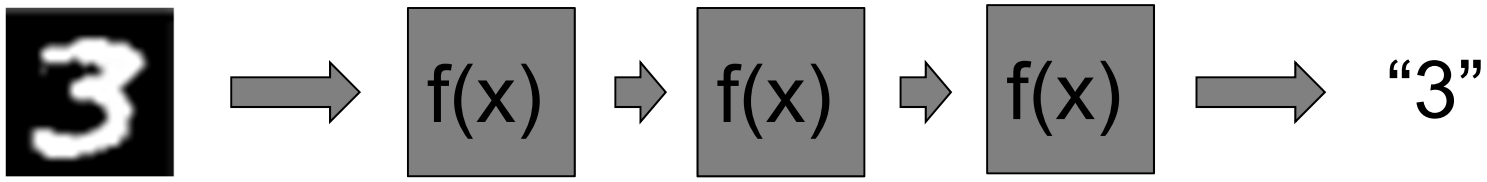
But:

- "**Enough units**" can be a very large number. There are functions representable with a small, but deep network that would require exponentially many units with a single layer.

  (e.g., Hastad et al. 1986, Bengio & Delalleau 2011)

- The proof only says that a shallow network *exists*, it does not say how to find it. Evidence indicates that it is easier to train a deep network to perform well than a shallow one.

Popular Network Architectures



- **Largest design space** in employing deep learning: **Which** functions (or “**layers**”) to use, and **in which combination**?

- Two popular architectures:
  - Convolutional Neural Networks (CNNs)
  - Recurrent Neural Networks (RNNs)

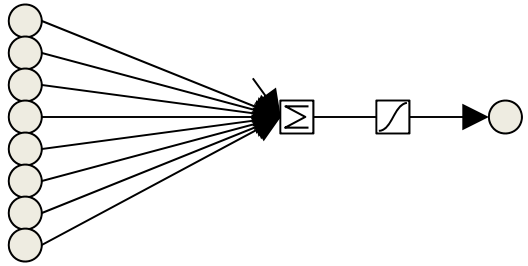
  that have a particular type of layers suited for particular problems (more later)

What are Artificial Neural Networks?

mathematical expressions, such as:

$y = \sigma(b + \mathbf{w}^T\mathbf{x})$ (equivalent to logistic regression)

expression can be visualized as a graph:



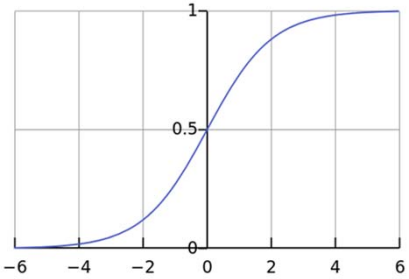
$\mathbf{x}$ $b + \mathbf{w}^T\mathbf{x}$ y

Output value is computed as a **weighted sum of its inputs,**

$$b + \mathbf{w}^T\mathbf{x} = b + \Sigma_i w_i x_i$$

followed by a nonlinear function:

sigmoid ->

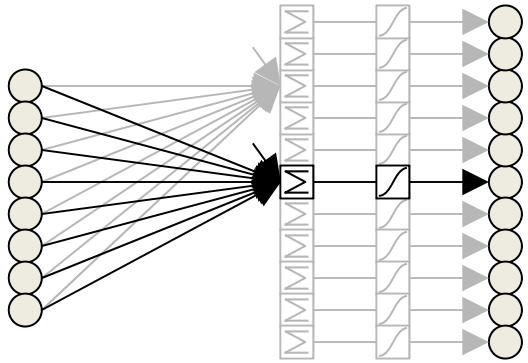


What are Artificial Neural Networks?

mathematical expressions, such as:

$$\mathbf{y} = \sigma(\mathbf{b} + \mathbf{W}^T\mathbf{x})$$ (multiple logistic regressions)

expression can be visualized as a graph:



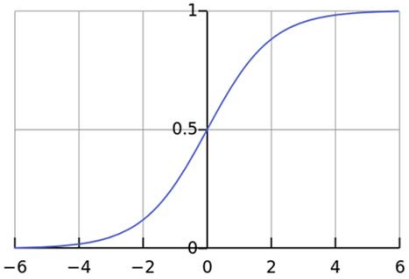
x **b + W^{T}x** **y**

Output values are computed as **weighted sums of their inputs,**

$$\mathbf{b} + \mathbf{W}^T\mathbf{x} = b_j + \Sigma_i w_{ij} x_i$$

followed by a nonlinear function:

sigmoid ->

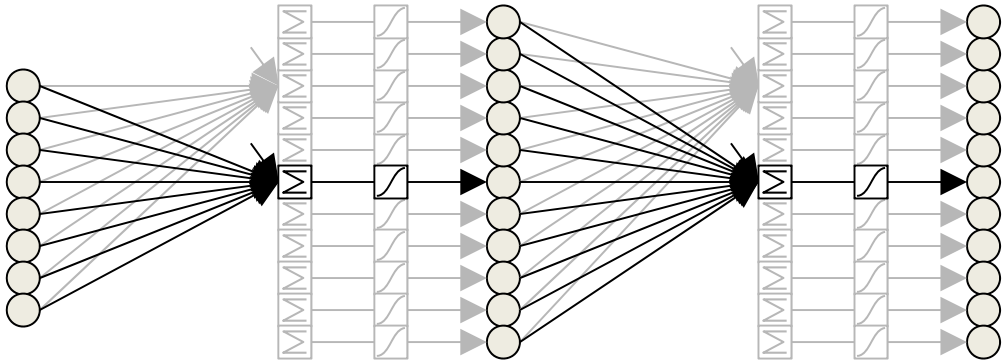


What are Artificial Neural Networks?

mathematical expressions, such as:

$$\mathbf{y} = \sigma(\mathbf{b}_2 + \mathbf{W}_2^T \sigma(\mathbf{b}_1 + \mathbf{W}_1^T \mathbf{x})) \qquad \text{(stacked logistic regressions)}$$

expression can be visualized as a graph:



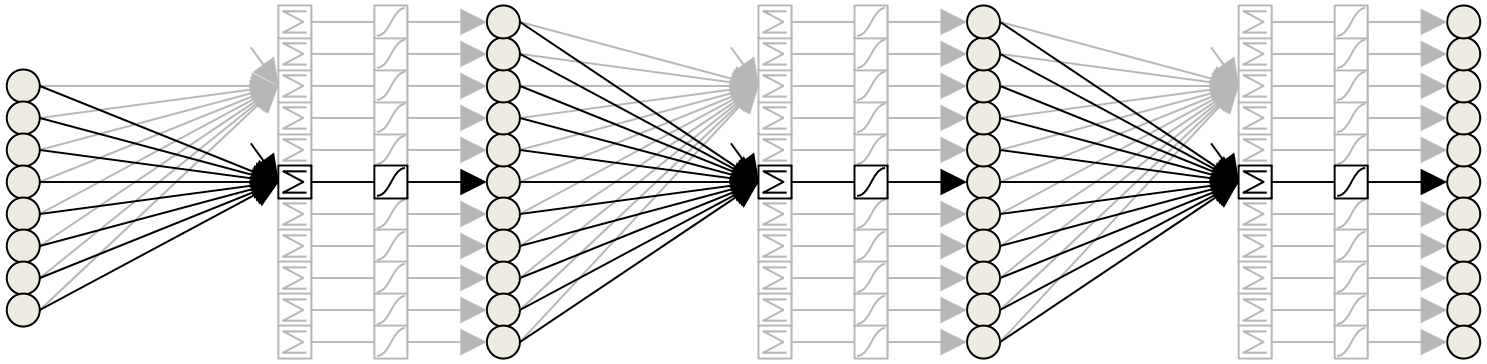
$\mathbf{x}$ $\mathbf{b}_1 + \mathbf{W}_1^T \mathbf{x}$ $\mathbf{h}$ $\mathbf{b}_2 + \mathbf{W}_2^T \mathbf{h}$ $\mathbf{y}$

What are Artificial Neural Networks?

mathematical expressions, such as:

$$\mathbf{y} = \sigma(\mathbf{b}_3 + \mathbf{W}_3^T\sigma(\mathbf{b}_2 + \mathbf{W}_2^T\sigma(\mathbf{b}_1 + \mathbf{W}_1^T\mathbf{x})))$$

expression can be visualized as a graph:



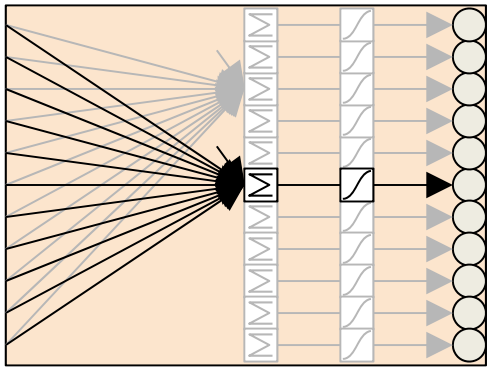
$\mathbf{x}$ $\mathbf{b}_1 + \mathbf{W}_1^T\mathbf{x}$ $\mathbf{h}_1$ $\mathbf{b}_2 + \mathbf{W}_2^T\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{b}_3 + \mathbf{W}_3^T\mathbf{h}_2$ $\mathbf{y}$

What are Artificial Neural Networks?

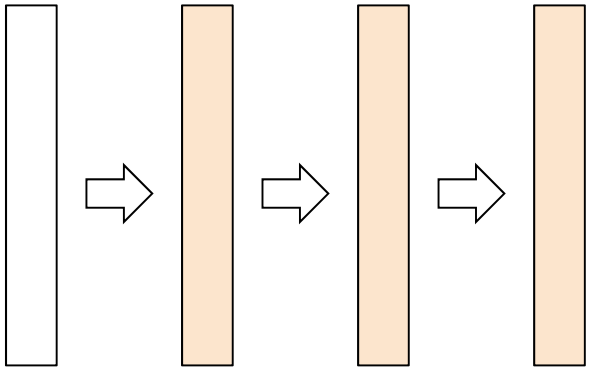
mathematical expressions, such as:

$$f_{\mathbf{W},\mathbf{b}}(\mathbf{x}) = \sigma(\mathbf{b} + \mathbf{W}^T\mathbf{x}) \qquad \mathbf{y} = (f_{W_3,b_3} \circ f_{W_2,b_2} \circ f_{W_1,b_1})(\mathbf{x})$$

expression can be visualized as a graph:



“dense layer”



composed of simpler **functions**, commonly termed “**layers**”

Mathematical Reasons for Going "Deep"

But:

- "**Enough units**" can be a very large number. There are functions representable with a small, but deep network that would require exponentially many units with a single layer.

  (e.g., Hastad et al. 1986, Bengio & Delalleau 2011)

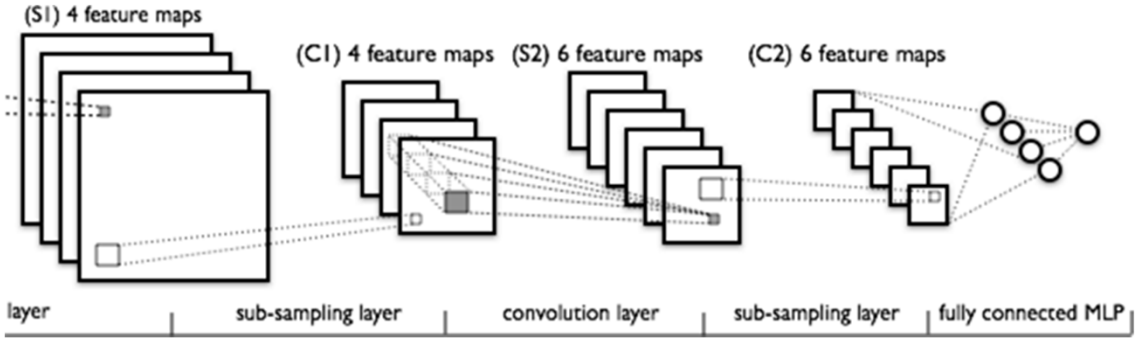
- The proof only says that a shallow network *exists*, it does not say how to find it. Evidence indicates that it is **easier to train a deep network to perform well than a shallow one.**

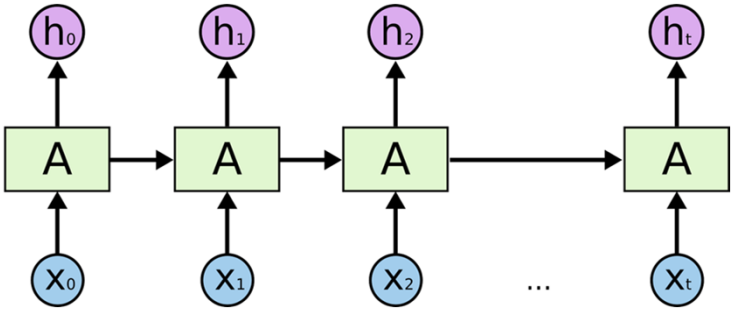
Modern
Neural Network Architectures

Neural Network Architectures

Two main Neural Network types in use today:

- Convolutional Neural Networks (ConvNets or CNN)
- Recurrent Neural Networks (RNN, LSTM, GRU)





CNNs vs. RNNs

Convolutional Networks:

- input is fixed-sized tensor (e.g. an **image**)
- produce a fixed-sized vector as output (e.g. probabilities of different classes)

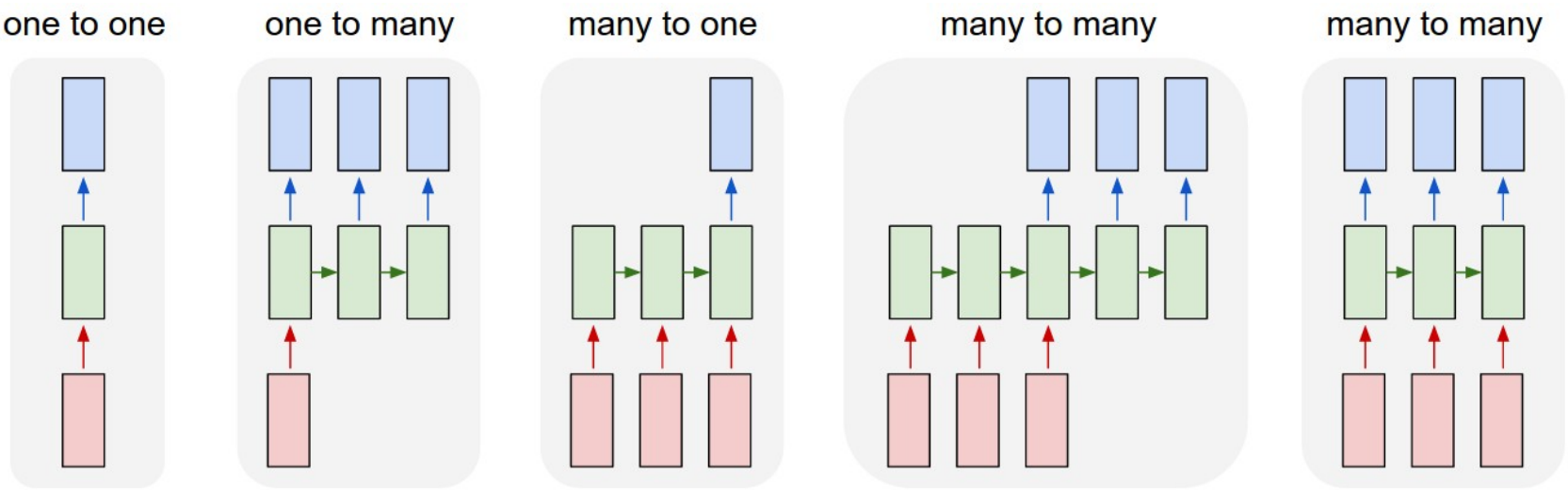
Recurrent Neural Networks:

- operate over **sequences** of vectors or tensors
- Example Applications: Text translation, Text to Speech, Speech to Text, etc.

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

Recurrent Neural Networks

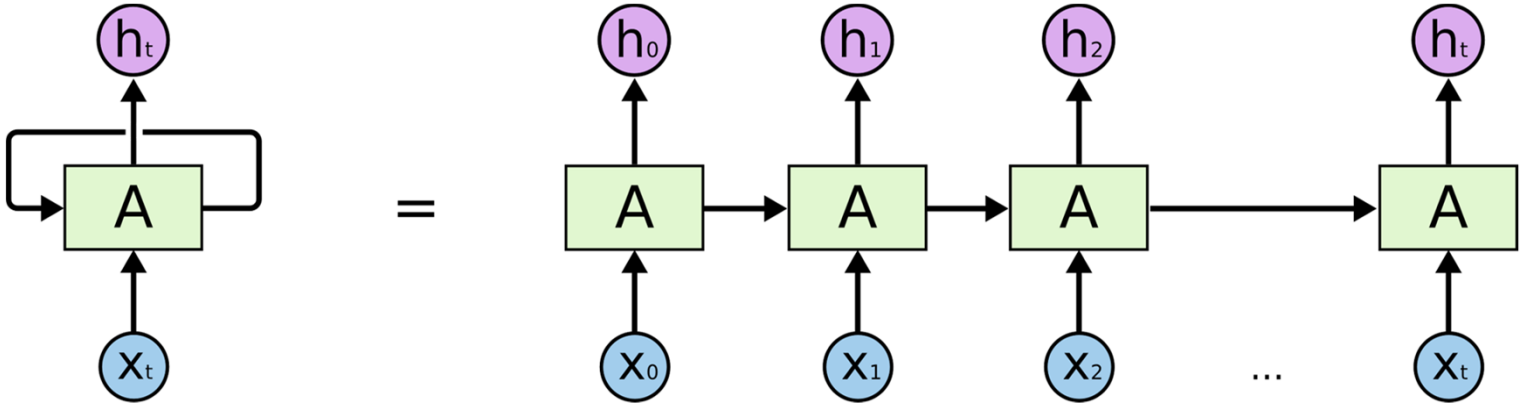
Process **sequences** of data in various ways:



Useful for time series data: text, sound, video, …

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

Recurrent Neural Networks (RNNs)



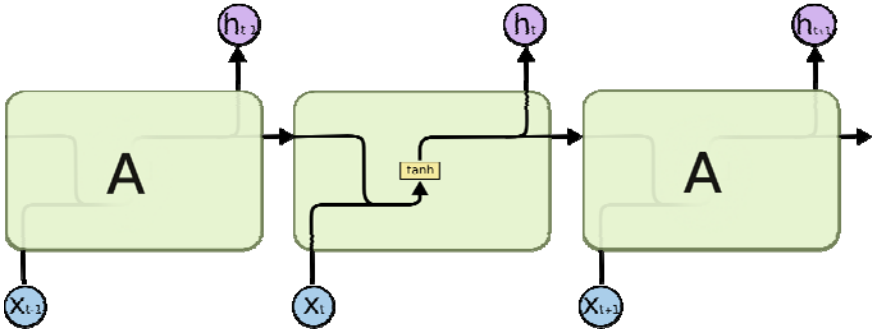
- RNNs process **sequences** of same-sized elements
- **Recurrent layers** process an input element together with **their own output** from the previous element

- Can be seen as a deep network with shared weights, unrolled in time: depth = length of sequence

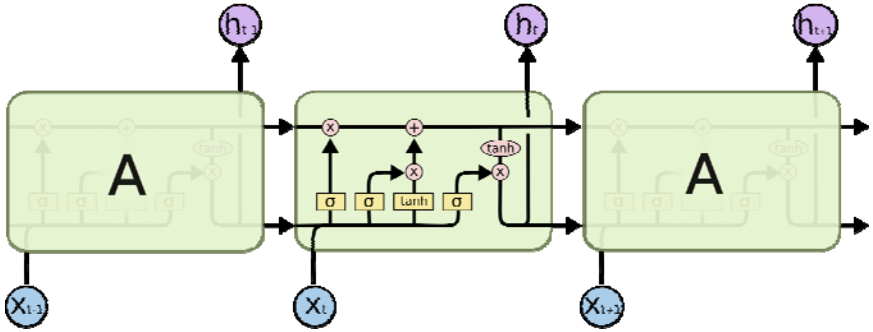
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN Variants: LSTM

- LSTM: Long Short-Term Memory
- can store longer term memory
- thus learns better from experience



Standard RNN



LSTM
contains 4 interacting layers

Details: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM: Hochreiter and Schmidhuber 1997

RNN Variants

- LSTM: Long Short-Term Memory
- GRU: **Gated Recurrent Units**
  - gating mechanism for RNNs (2014)
  - Similar performance but fewer parameters than LSTM
- Neural Turing Machine

- All types of RNNs and CNNs can also be combined (e.g. an RNN processing sequences of CNN outputs)

Deep Learning Architectures for Music Analysis

Convolutional Networks:

- when analyzing audio spectrogram excerpts
- when time sequence does not necessarily play important role (i.e. processing audio samples as a whole; e.g. for genre, mood recognition, …)

Recurrent Neural Networks:

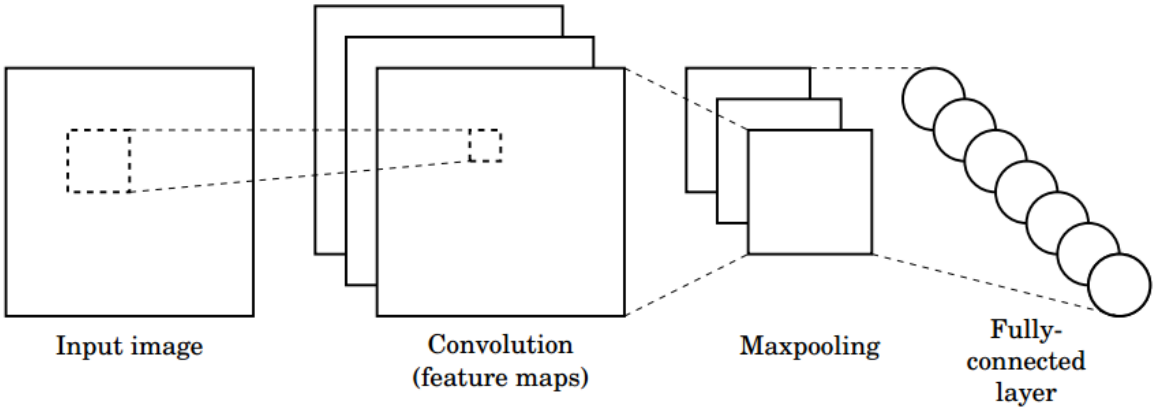
- when sequence and time series are important (e.g. (genre), melody, beat onset detection)

➔ often a mix of CNNs with RNNs is used

Recurrent Neural Networks

we will not go into further detail about RNNs in this tutorial

Convolutional Neural Network (CNN)



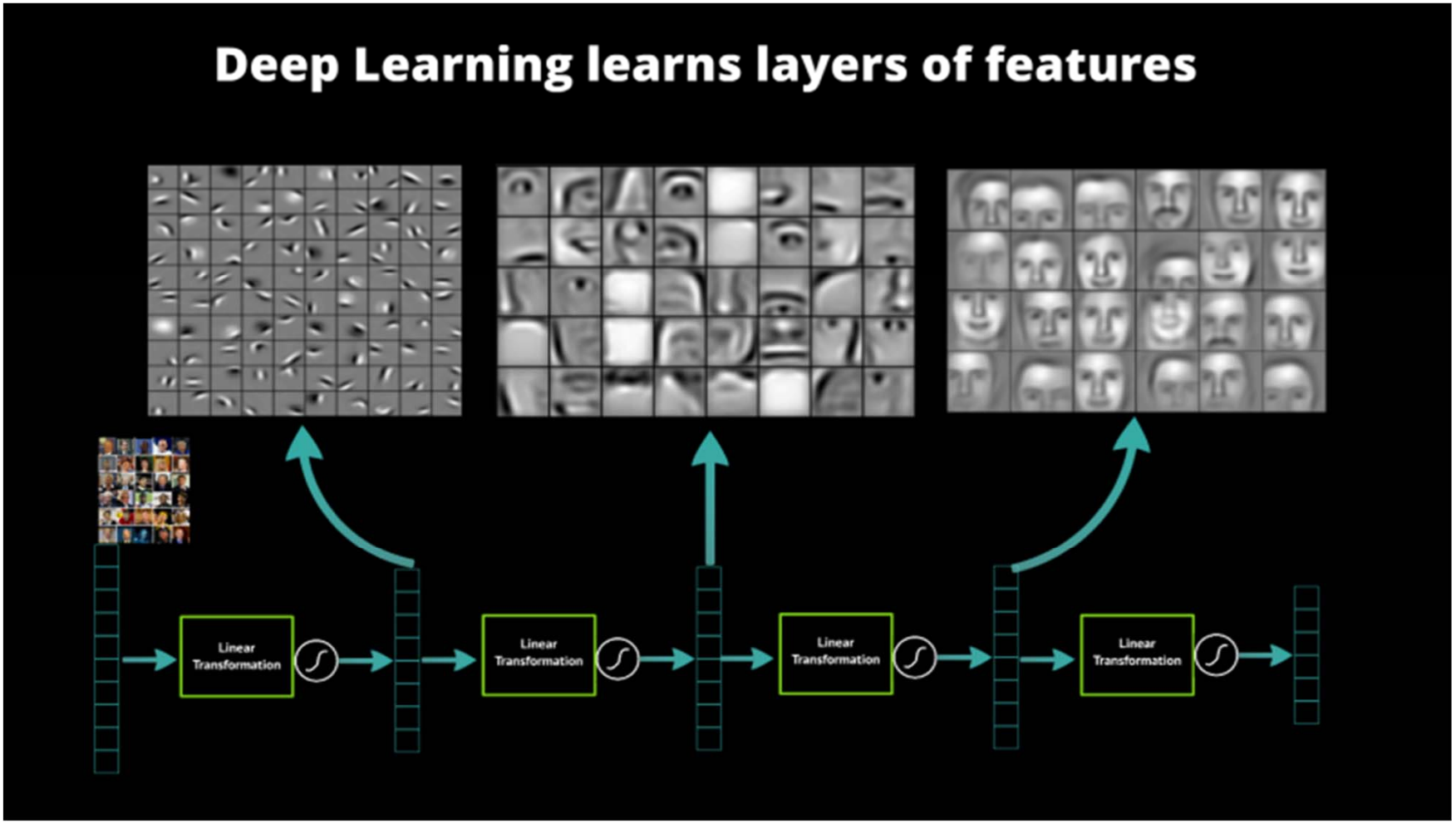
Combines three types of layers:

- **Convolutional layer:** performs 2D convolution of 2D input with multiple learned 2D kernels – learns shapes
- **Subsampling layer:** replaces 2D patches by their maximum ("max-pooling") or average ("average-pooling") – reduces resolution
- **Fully-connected layer:** computes weighted sums of its input with multiple sets of learned coefficients – maps to output

Applies a *nonlinear* function after each linear operation (without, a deep network would be linear despite its depth).

Img: https://www.vaetas.cz/blog/intro-convolutional-neural-networks/

Convolutional Neural Networks (CNN)



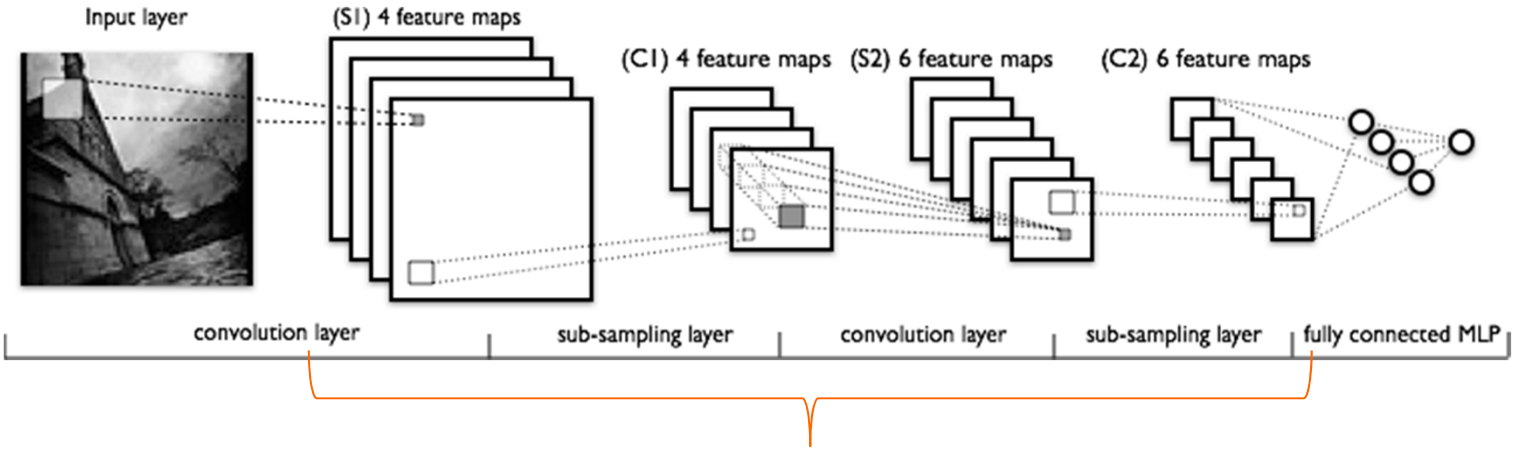
Note: the images are conceptual here and do not represent the actual output of the neurons.

Motivation for CNNs

- Humans organize their ideas and concepts **hierarchically**
- Humans **first** learn **simpler concepts** and then compose them to represent more abstract ones
- Engineers **break-up solutions into multiple levels** of abstraction and processing
- It would be good to automatically learn / discover these concepts

Y. Bengio, Deep Learning, MLSS 2015, Austin, Texas, Jan 2014

Convolutional Neural Network (CNN)



Convolution + Pooling repeat and alternate in deeper networks

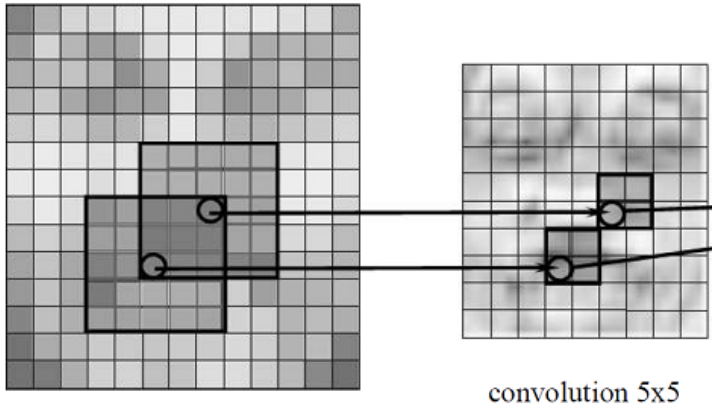
Combines three types of layers:

- **Convolutional layer:** performs 2D convolution of 2D input with multiple learned 2D kernels – learns shapes
- **Subsampling layer:** replaces 2D patches by their maximum ("max-pooling") or average ("average-pooling") – reduces resolution
- **Fully-connected layer:** computes weighted sums of its input with multiple sets of learned coefficients – maps to output

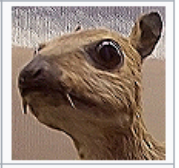
Img: http://deeplearning.net/tutorial/lenet.html

What is a Convolution?

- Apply local filter kernels and slide them over the input
- Instead of using predefined kernels, these kernels are the neurons that are learned!



convolution 5x5

Operation	Kernel	Image result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

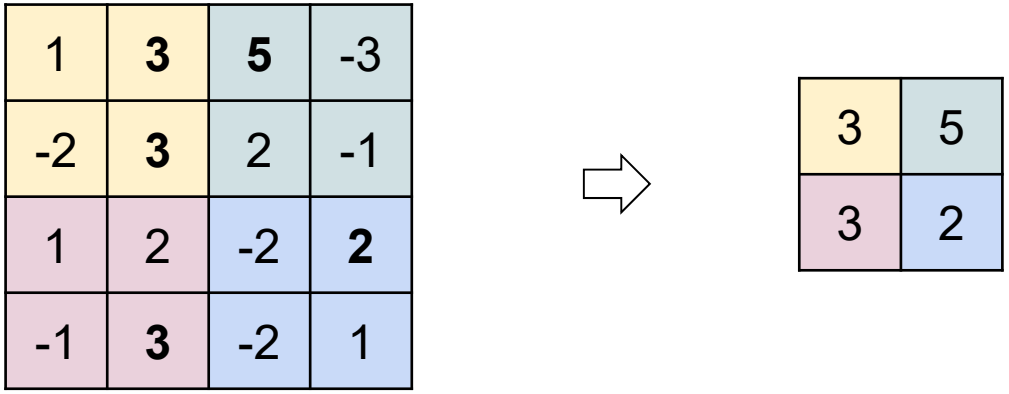
Images: http://sanghyukchun.github.io/75/
https://en.wikipedia.org/wiki/Kernel_(image_processing)

What is Pooling?

Second very important aspect of a CNN:

(also called subsampling or downsampling)

A **pooling layer** reduces the size of feature maps (i.e. output of a CNN layer and thus the input to the next layer)



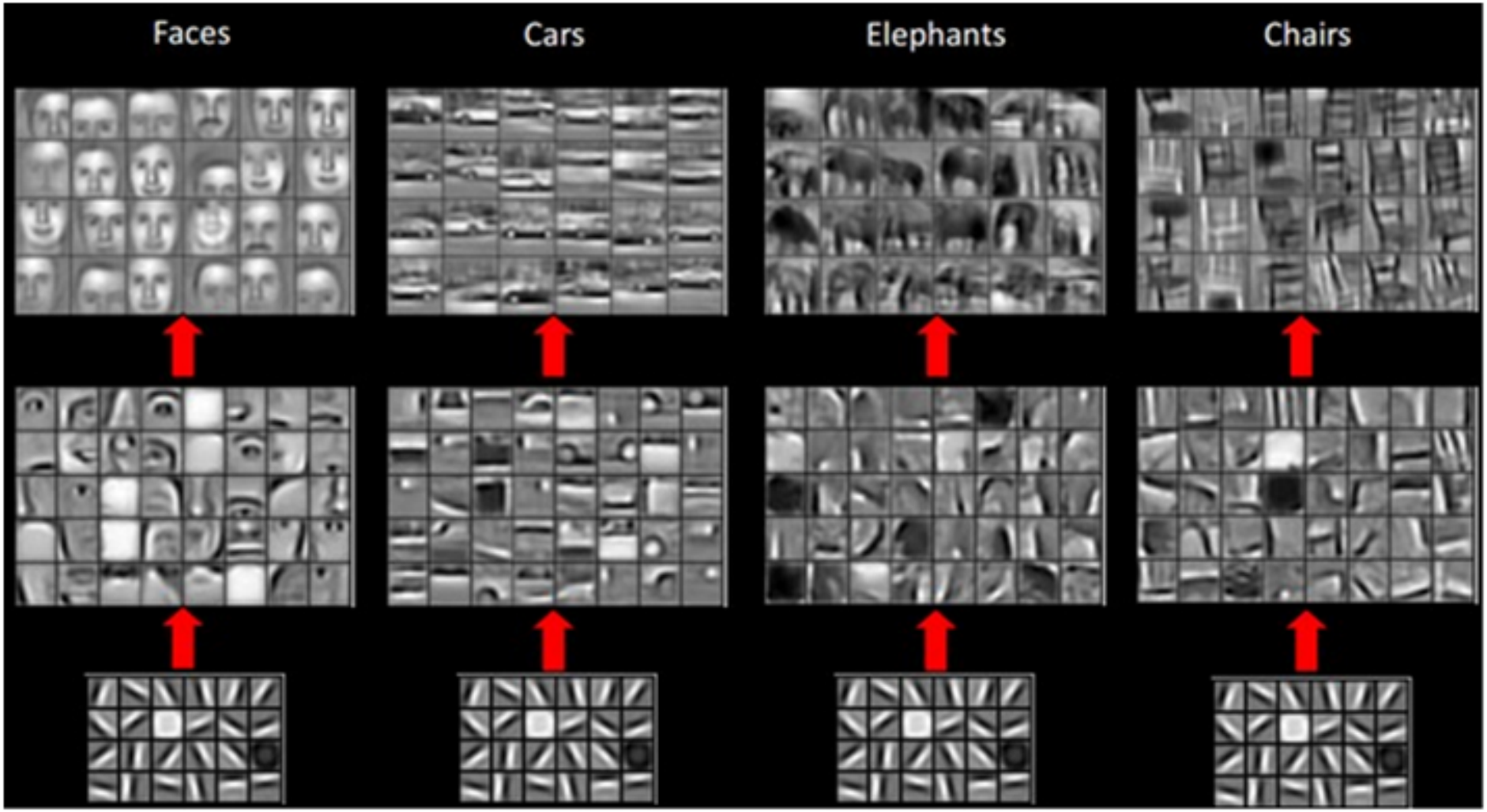
Max pooling: take the max. activation across small regions

(e.g. 2x2, as in the example above)

it can also be considered as an aggregation step

Image Object Recognition:

Deep Neural Nets always follow the same principles

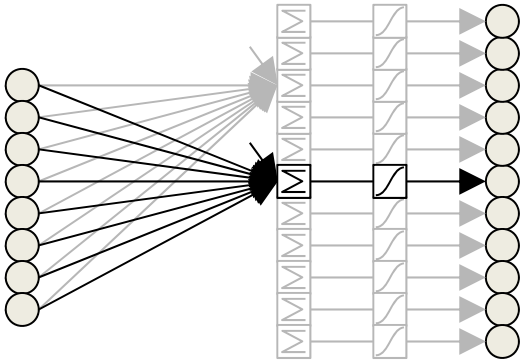


Src: http://stats.stackexchange.com/questions/146413/why-convolutional-neural-networks-belong-to-deep-learning

Full vs. Convolutional Layer / Network

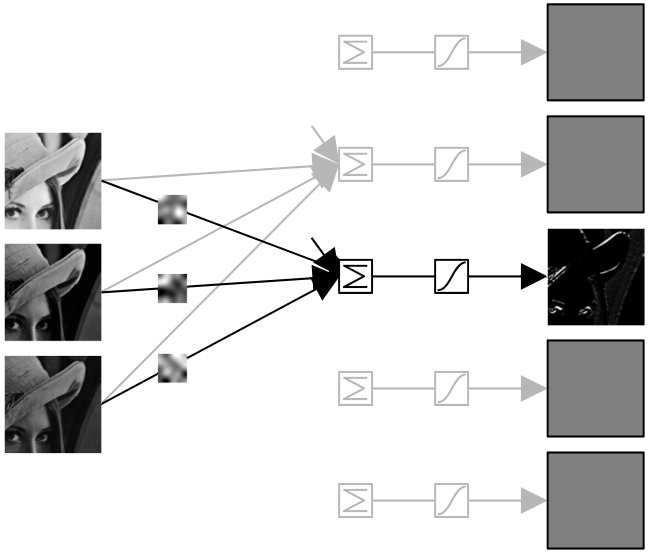
Fully-connected layer:

Each **input** is a **scalar** value,
each **weight** is a **scalar** value,
each output is the sum of
inputs **multiplied** by weights.



Convolutional layer:

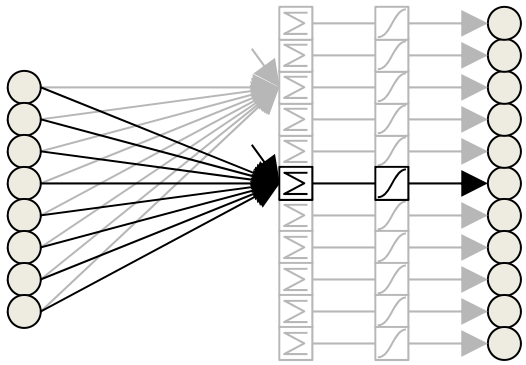
Each **input** is a **tensor** (e.g., 2D),
each **weight** is a **tensor**,
each output is the sum of
inputs **convolved** by weights.



Full Layer

Fully-connected layer:

Each **input** is a **scalar** value, each **weight** is a **scalar** value, each output is the sum of inputs **multiplied** by weights.



→ Swapping two inputs does not change the task.

We can just swap the weights as well

Example task:

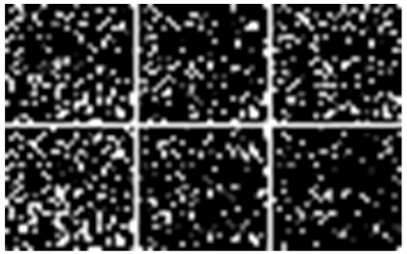
Distinguish *iris setosa*, *iris versicolour* and *iris virginica*

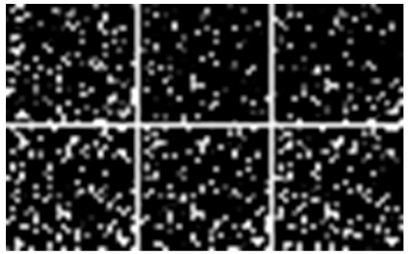
Input: (sepal length, sepal width, petal length, petal width)

Equivalent: (sepal width, petal length, sepal length, petal width)

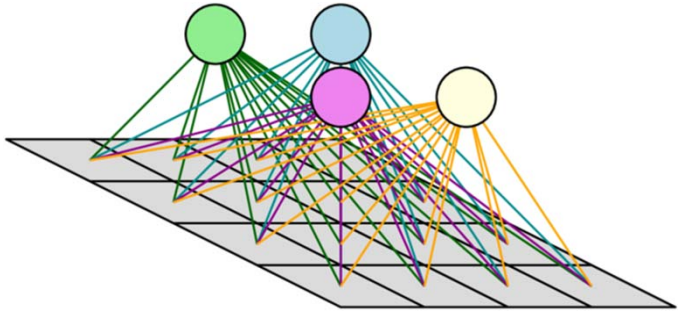
Full vs. Convolutional Layer / Network

Original input





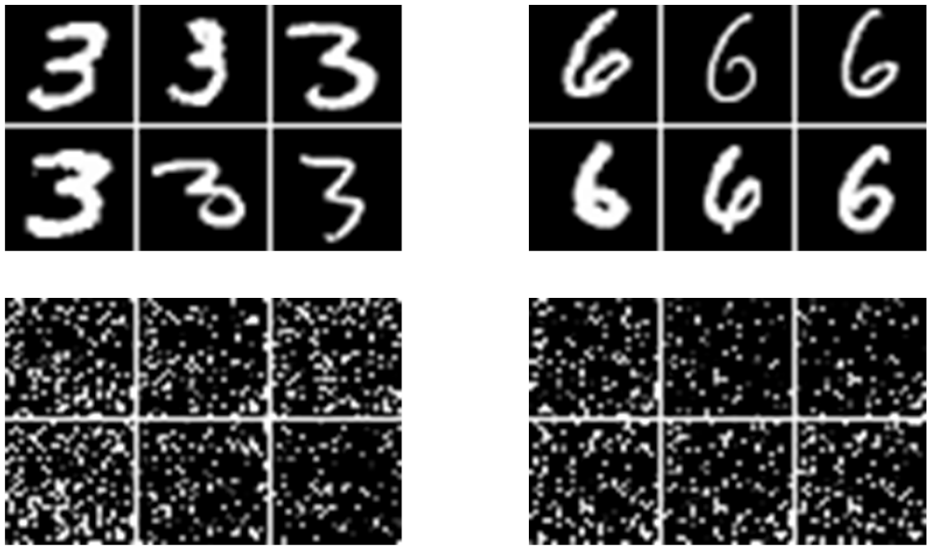
Permuted input



In a **fully**-connected layer, **every** input is connected with the following layer.

Consequently, the **order does not matter!**

Motivation for Convolutions



Original input

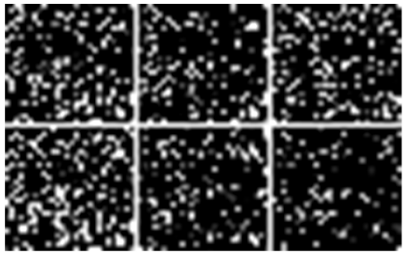
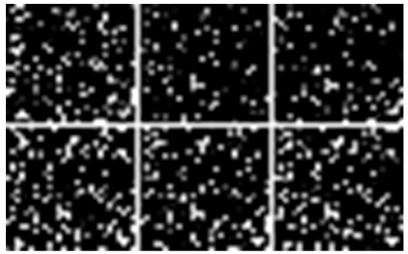
Permuted input

For a fully-connected network, the task is the same.

For humans, in the permuted version it is outright **impossible** to recognize the numbers!

Motivation for Convolutions


Original input

Permuted input

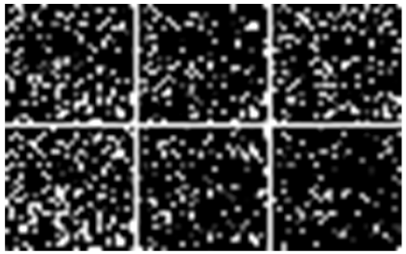
Humans make use of the **spatial layout** of visual data.

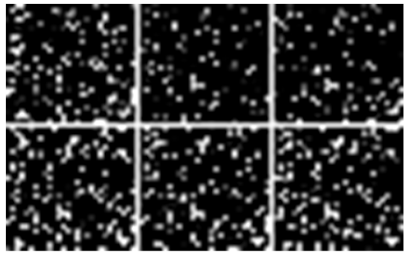
We (our eyes) do not see a full image at once, but only a **local neighbourhood** around a focal point.

→ It may be useful to connect each artificial neuron to a **small part of the input** image only.

Motivation for Convolutions

Original input





Permuted input

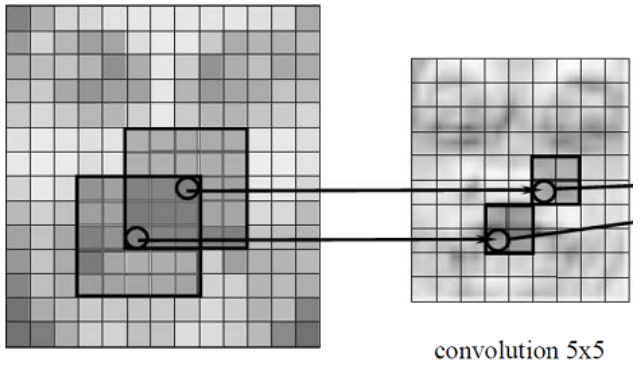
Humans reuse “feature detectors”:

We do not have separate eyes to look at the upper and lower part of an image. We apply the same **local feature detectors** to different positions.

→ It may be useful to **share weights** between neurons applied to different parts of the input image.

Convolutions: Conclusions

Operation is equivalent to convolution with a small kernel.

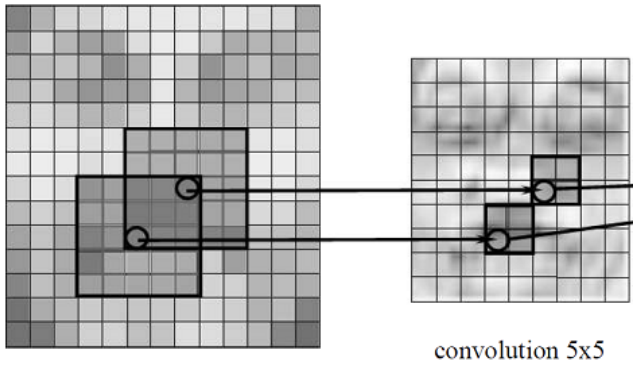


Consequences:

- Input **permutation <u>does</u> make a difference** now
- Also for the next layer: **Spatial layout is preserved**
- Can process large images with **few learnable weights**
- Weights are required to be applicable over the full image.

Convolutions: Conclusions

Operation is equivalent to convolution with a small kernel.

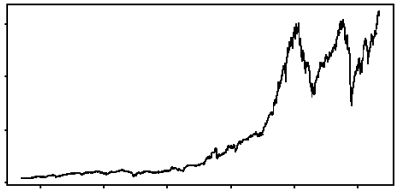


Advantages:

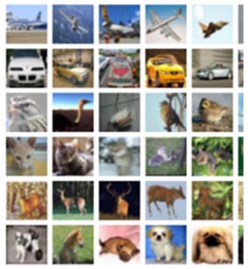
- Fewer parameters
- Faster learning
- Strong regularizer
- Learns spatial artefacts, i.e. repeating objects
- Less prone to overfitting!

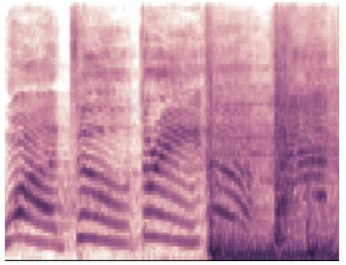
Convolutions in 1D, 2D, 3D

1D: time series,
audio waveforms

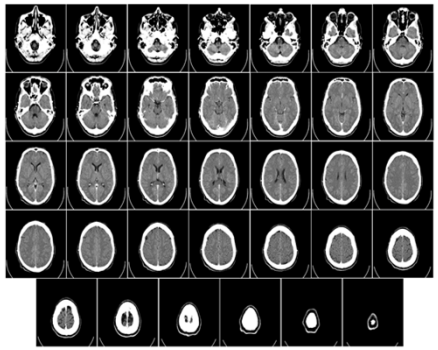


2D: images,
audio spectrograms

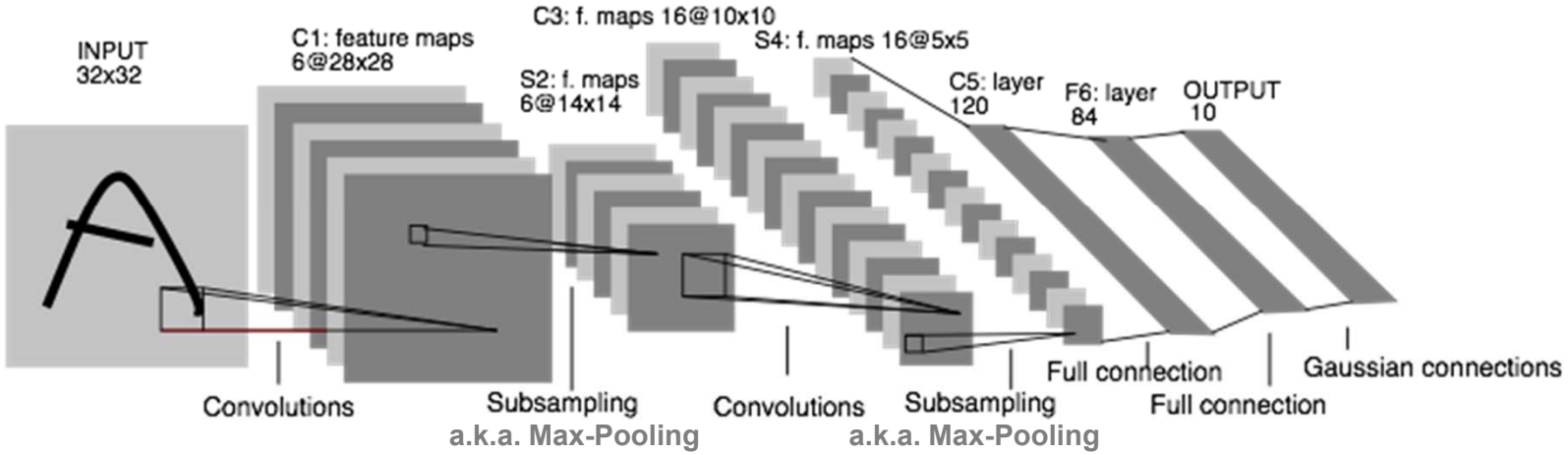




3D: volumetric data,
video



Convolutional Neural Network:
A complete architecture example



- **Convolutional layers**: local feature extraction
- **Pooling layers**: data reduction + some translation invariance
- **Fully-connected layers**: integrate information over full input, produce output

Convolutions: Kernel sizes

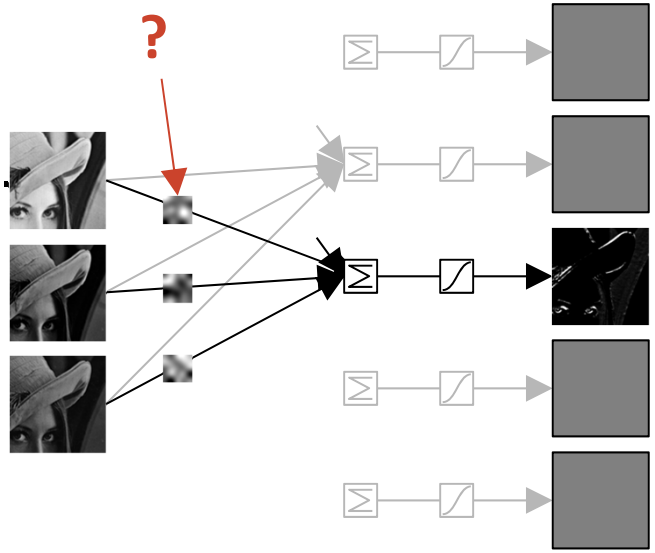
How big to make the learnable kernels?

3x3 kernel: Can only see small structure.

7x7 kernel: Sees more, but more weights.

224x224 kernel: Possibly sees full input.

Equivalent to fully-connected then.



Sweet spot somewhere in between?

Convolutions: Kernel sizes

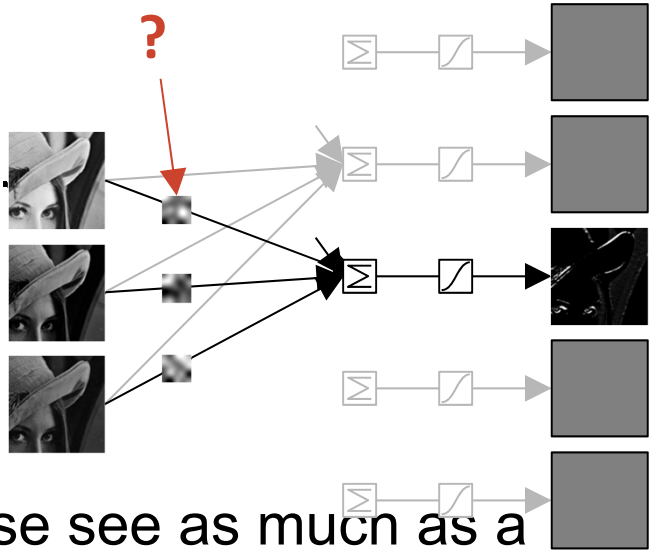
How big to make the learnable kernels?

3x3 kernel: Can only see small structure.

7x7 kernel: Sees more, but more weights.

224x224 kernel: Possibly sees full input.

Equivalent to fully-connected then.



Stack of 3x3 convolutions: Three of those see as much as a 7x7 convolution, with 45% fewer weights (27/49) and two additional nonlinearities. Win-win situation!

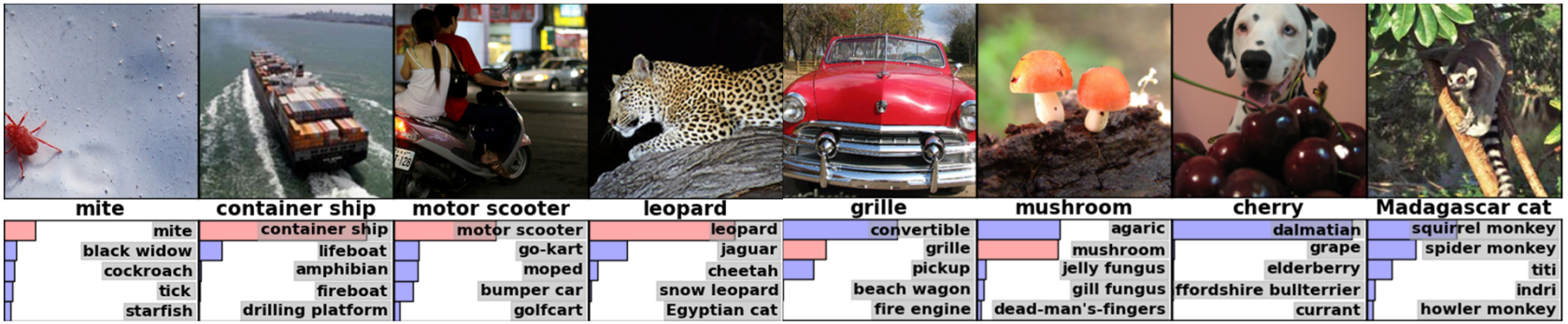
→ **Motivation for going deeper!**

Application Domains of CNNs

Image Object Classification

IMAGENET Large Scale Visual Recognition Challenge:

1.2 million training images of 1000 classes



- 2012: AlexNet 16.4% top-5 error (first deep net) (2nd best: 26.2%)
- 2013: Clarifai: 11.7% top-5 error, or 11.2% with extra data
  (most contestants used deep nets now).
- 2014: GoogLeNet: 6.7%. Andrej Karpathy (a human): 5.1%
- 2015: ResNets: 3.6%
- 2016: 2.9% Ensemble of Inception, Inception-Resnet, ResNet and Wide Residual Network

Neural Style Transfer

Starting point: CNN trained for object classification

Use network in reverse: Find image that matches high-level representation of photograph and low-level repr. of painting

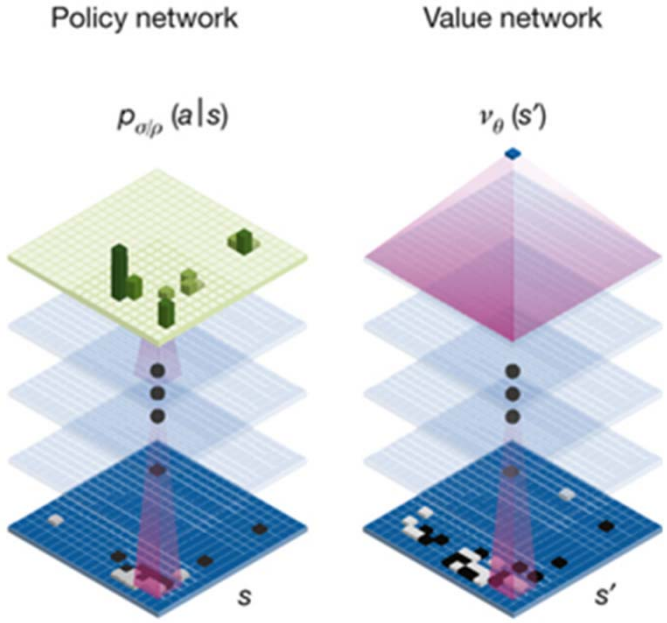
Aug 2015: A Neural Algorithm of Artistic Style, http://arxiv.org/abs/1508.06576

AlphaGo

Input: Go board (stone positions and some helping markers)

Output: position for next move (policy) or likely winner (value)

CNN combined with Monte-Carlo Tree Search for playing



Nature 526, Jan 2016: Mastering the game of Go with deep neural networks and tree search

Speech Recognition

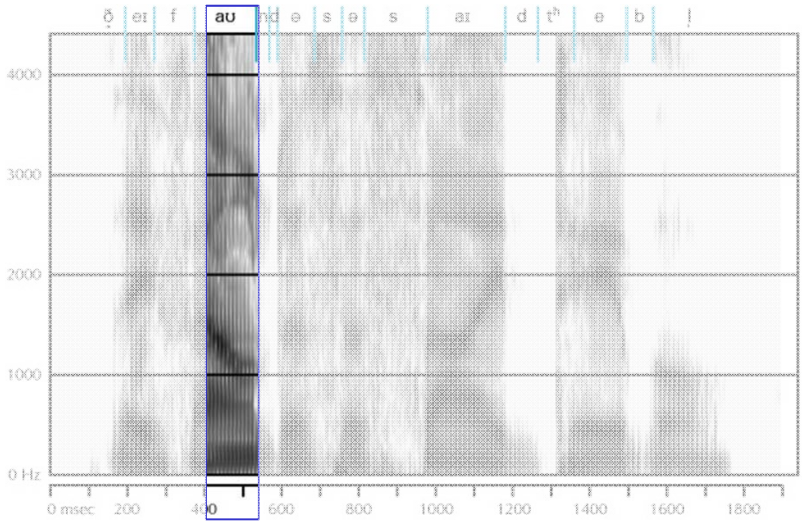
First successful networks in speech recognition:

Input: short spectrogram excerpt

Output: detected phoneme

Method: CNN

Has to be combined with word and language model, requires carefully-aligned training data.



→ aʊ

Dec 2015: Deep Speech 2: End-to-End Speech Recognition in English and Mandarin, http://arxiv.org/abs/1512.02595

Speech Recognition

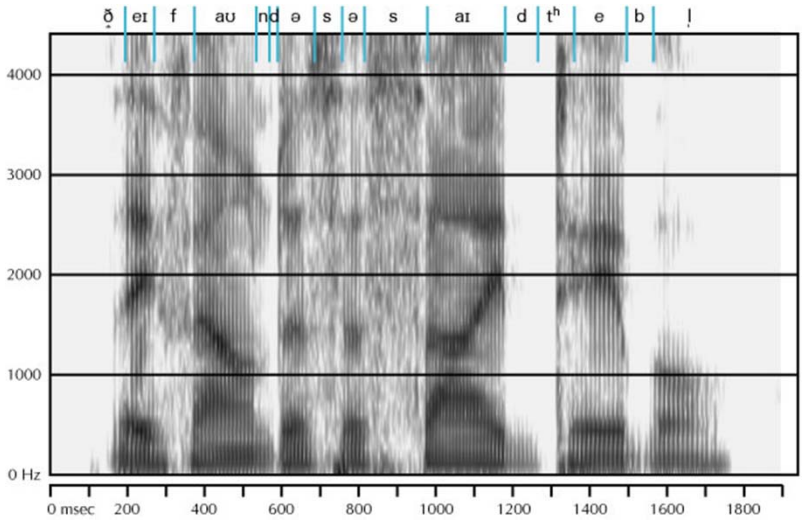
State of the art truly learns **from end to end**:

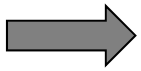
Input: spectrogram of recorded sentence

Output: transcribed sentence as a sequence of characters

Method: CNN for initial processing, RNN for temporal context

Works with coarsely aligned training data, learns word and language model on its own, **learns English and Mandarin with the same architecture**.





"They found us a side table."

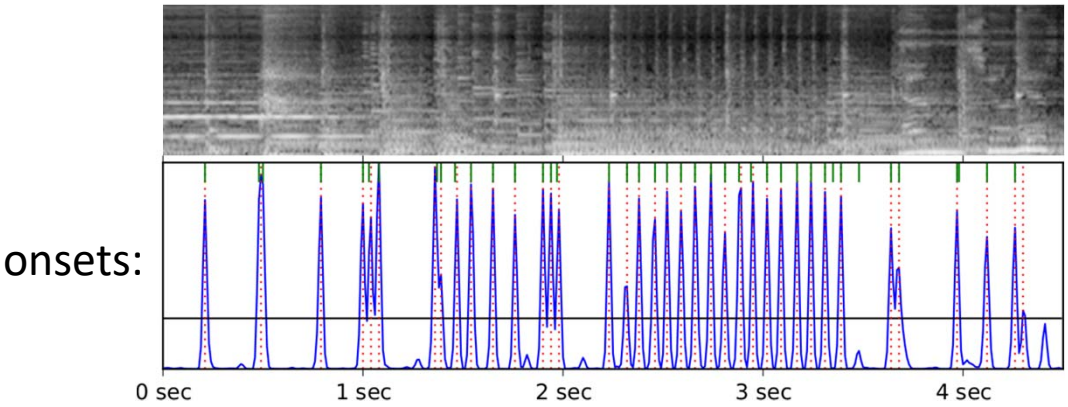
Dec 2015: Deep Speech 2: End-to-End Speech Recognition in English and Mandarin, http://arxiv.org/abs/1512.02595

Music Analysis: Note Onset Detection

Input: short spectrogram excerpt

Output: whether a note starts at center of excerpt

Combined result: starting positions of all notes played or sung



by Jan Schlüter

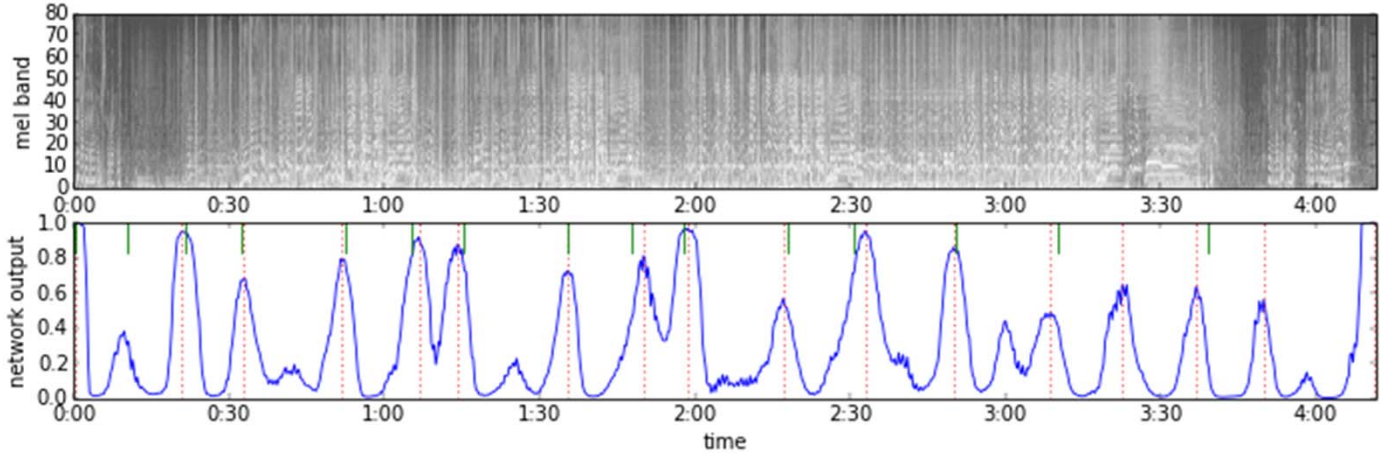
ICASSP 2014: Improved Musical Onset Detection with Convolutional Neural Networks

Music Analysis: Structural Segmentation

Input: long spectrogram excerpt

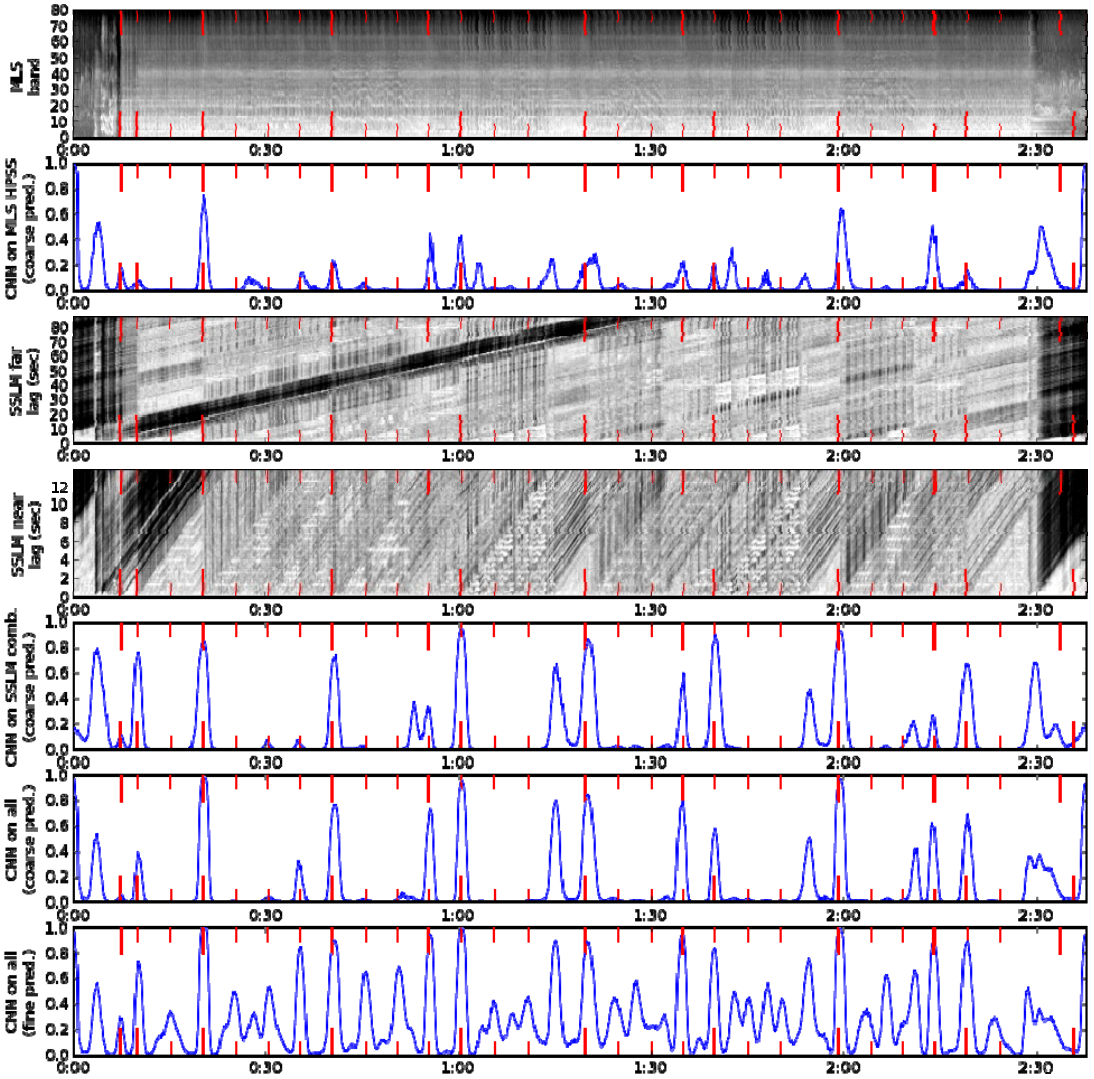
Output: whether music changes at center of excerpt

Combined result: positions of all structural boundaries



by Jan Schlüter

Music Analysis: Structural Segmentation



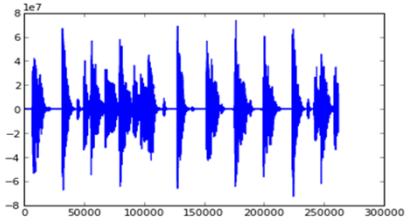
by Jan Schlüter

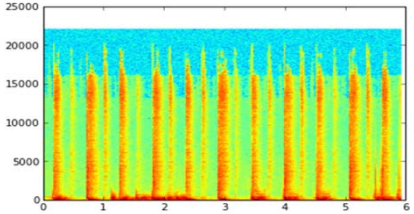
ISMIR 2015: Music Boundary Detection Using Neural Networks on Combined Features and Two-Level Annotations. http://ofai.at/research/impml/projects/audiostreams/ismir2015/

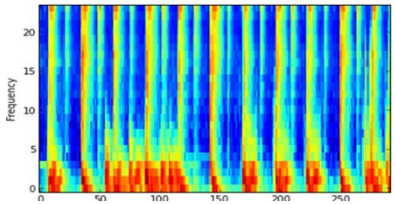
Music / Speech Classification

Distinguish audio excerpts between music and speech

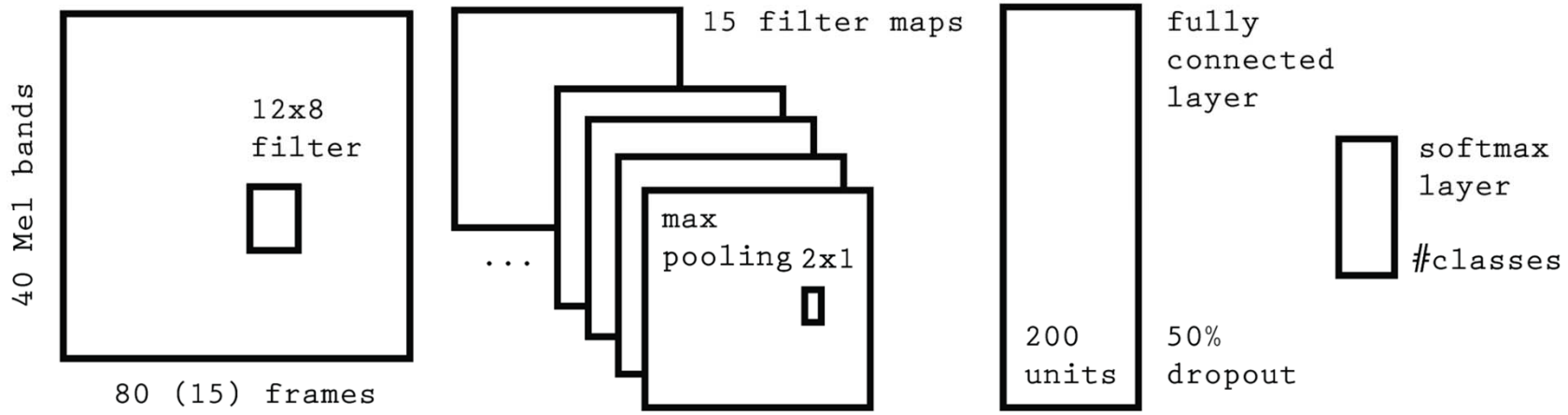
1. Pre-Processing: Waveform → Spectrogram → 40 Mel bands → Log scale







2. CNN with 1 layer, 15 filter maps + 1 full layer (input: 15 40x80 frames per file)

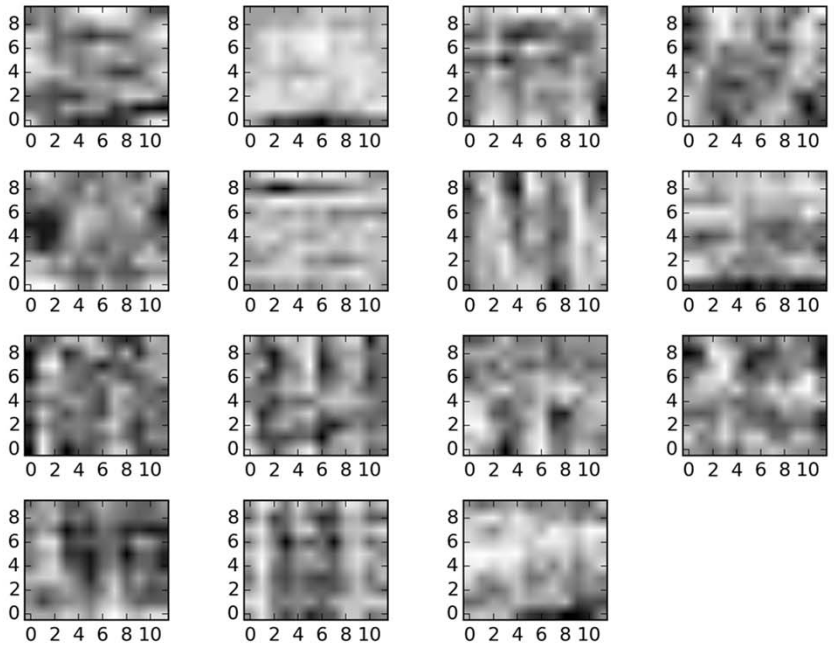


Winning algorithm **MIREX 2015** music/speech classification task **(99.73%) by Thomas Lidy**

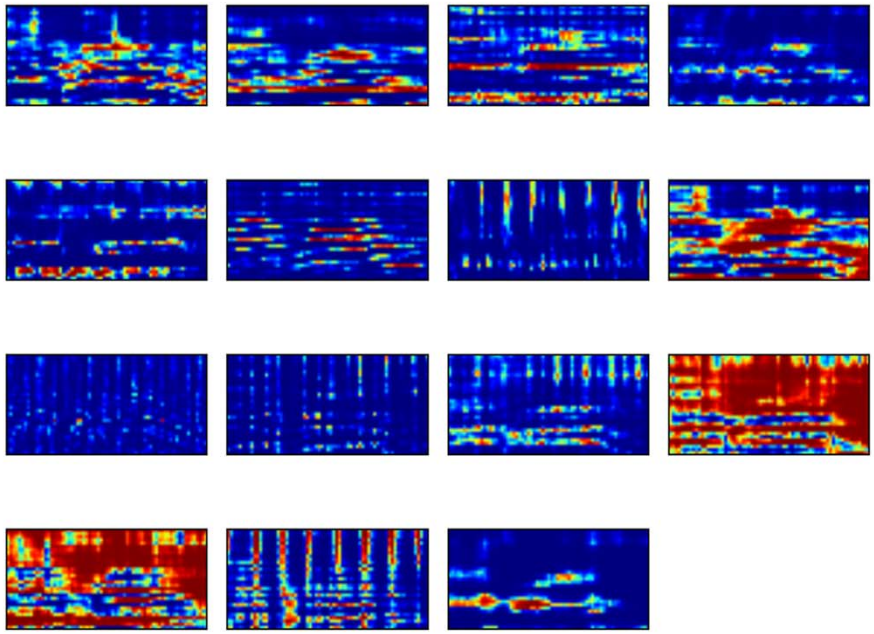
Visualizing CNN Filters

learned for Music/Speech Classification

Learned Filter Weights

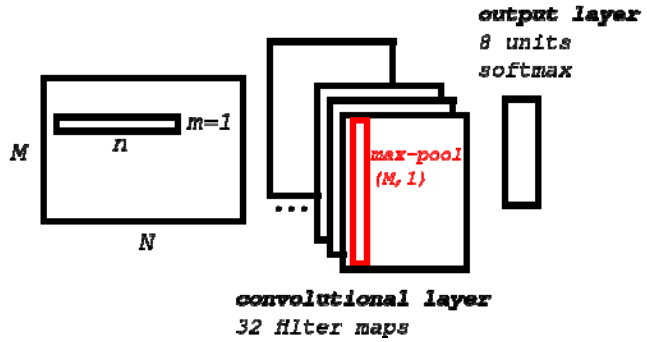


Convolved Spectrograms

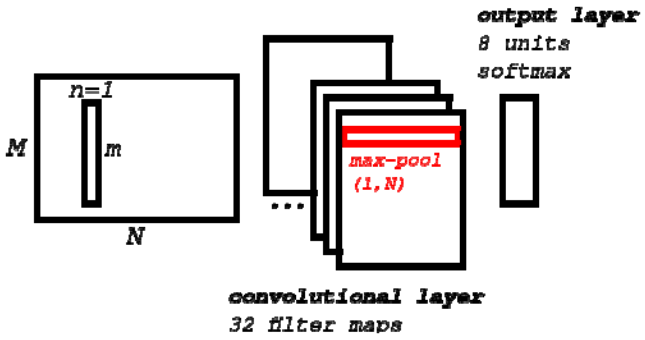


Musically Motivated CNN

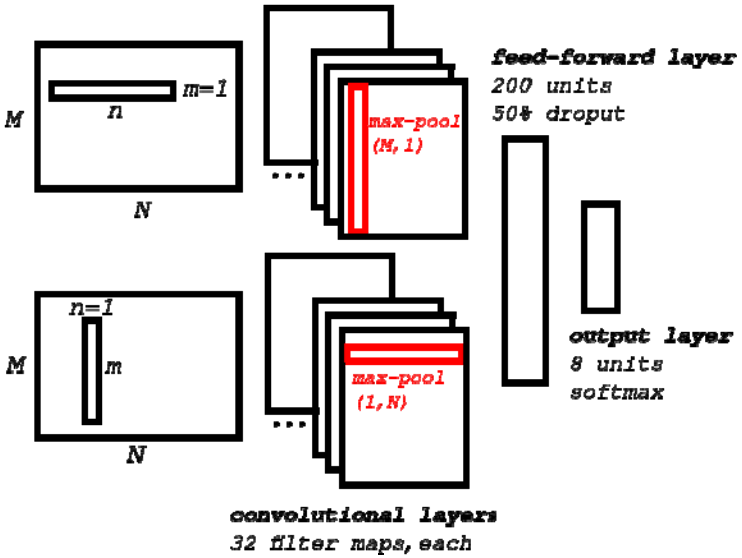
Time filter



Frequency Filter

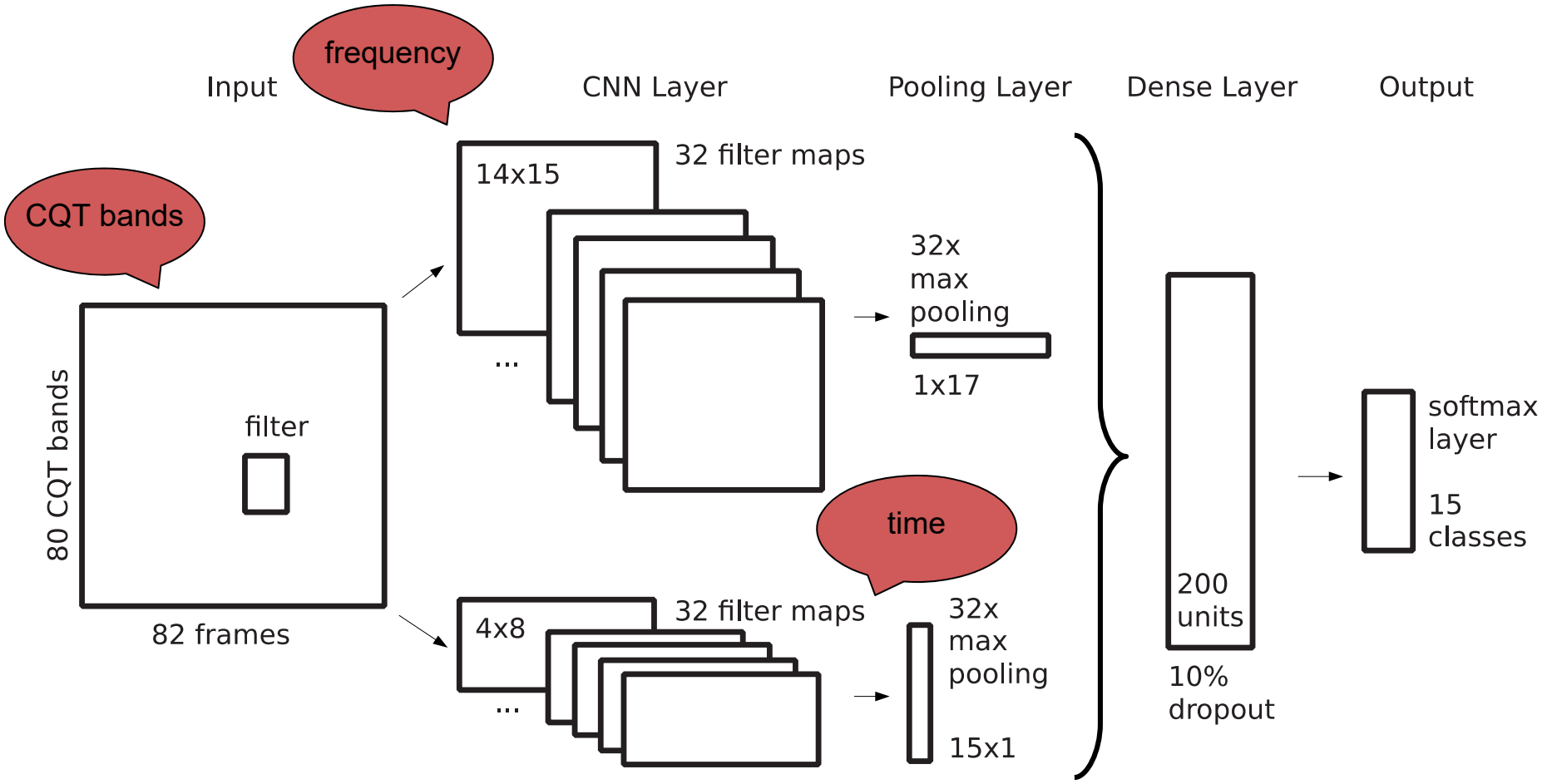


Time + Frequency Model



Jordi Pons, T. Lidy (CBMI 2016)

Parallel Architectures



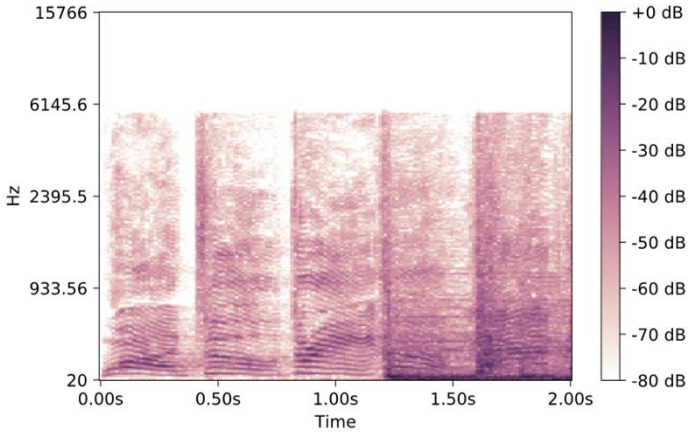
Winning model of DCASE 2016 Domestic Audio Tagging challenge
(child speech, male, female, video-game/TV, percussive sounds, other)

Lidy and Schindler (DCASE 2016)

DCASE 2016

Spectrogram vs. Mel bands vs. CQT

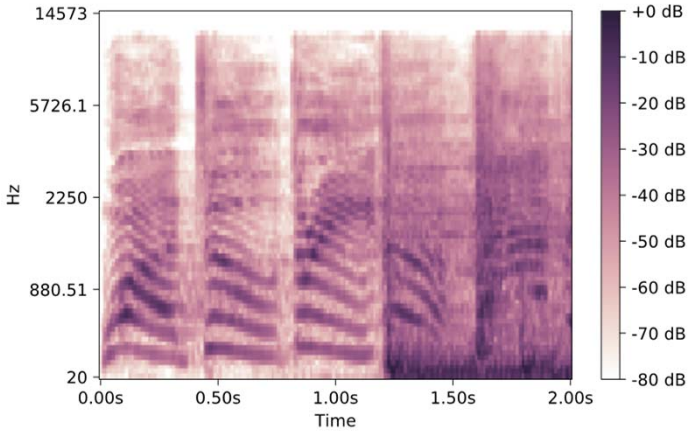
Spectrogram



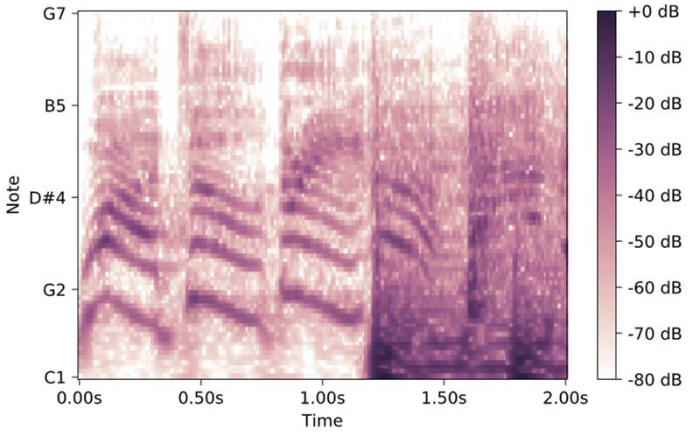
Mel vs. CQT

Transform	Bands/Frames	Acc max prob	Acc maj vote
Mel	40 x 80	76,23%	75,62%
Mel	80 x 80	76,55%	76,38%
CQT	80 x 82	**80,25%**	**80,07%**
CQT	84 x 82	78,11%	77,59%
CQT	126 x 82	79,39%	79,14%

Mel



CQT



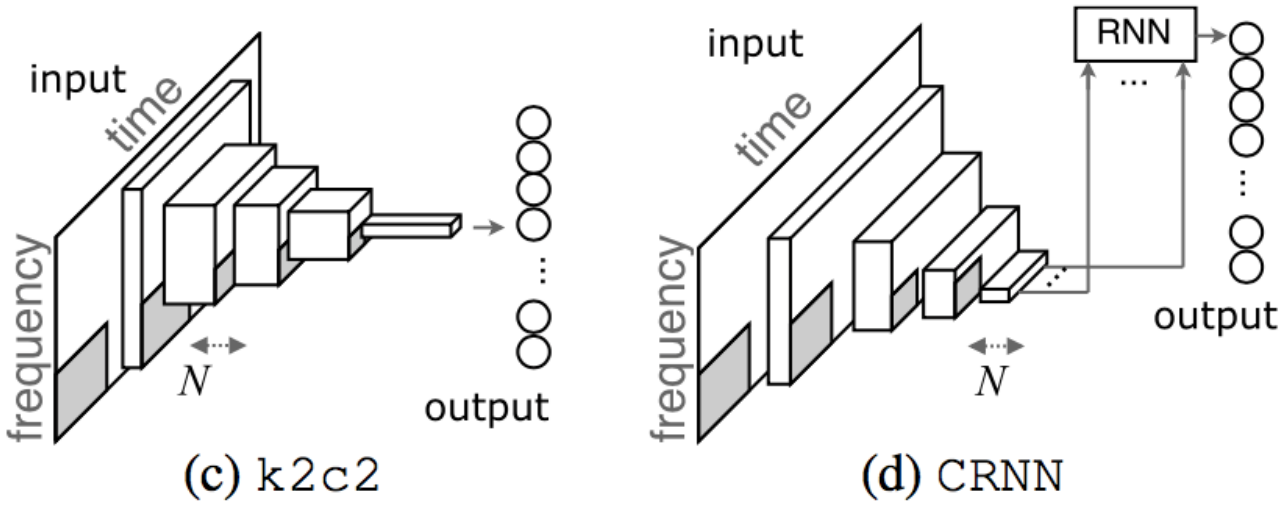
Lidy and Schindler (DCASE 2016)

CQT-BASED CONVOLUTIONAL NEURAL NETWORKS FOR AUDIO SCENE CLASSIFICATION

C-RNNs for Music Classification & Tagging

CNN

C-RNN

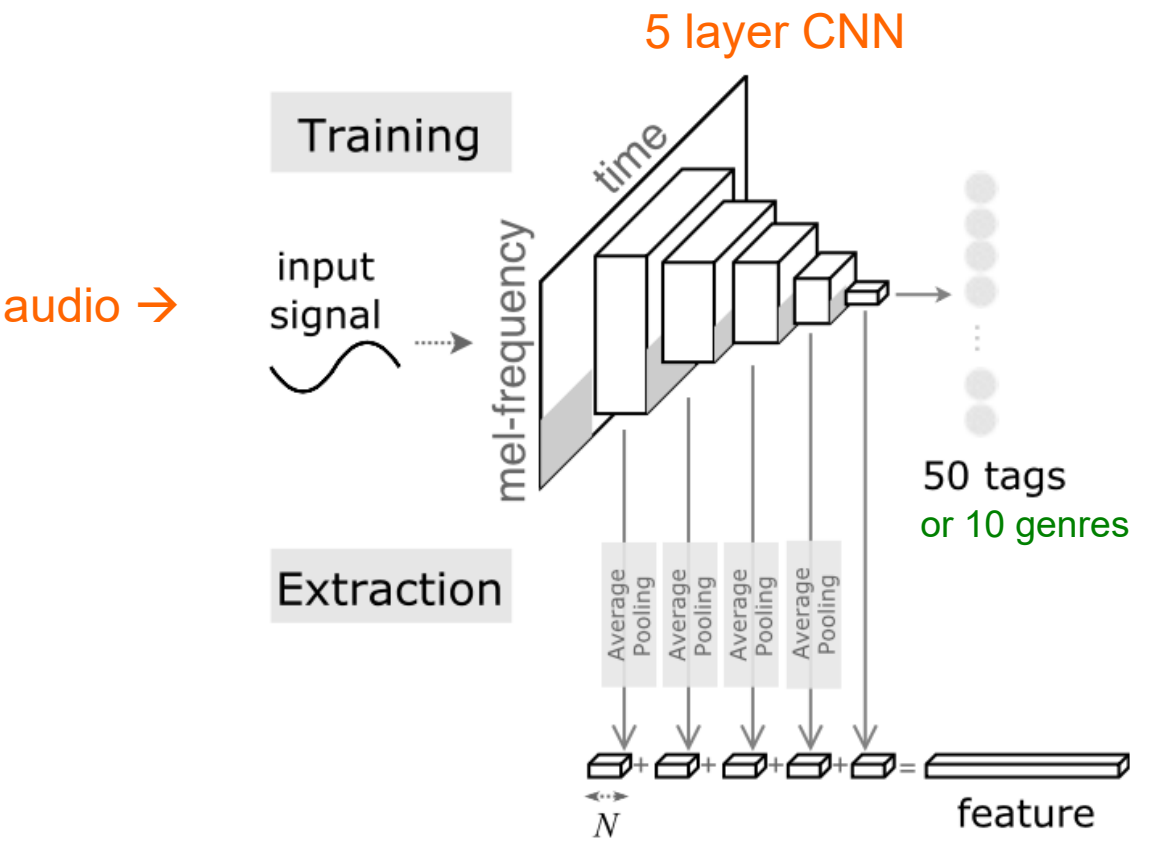


C-RNN (hybrid):

- CNNs for local feature extraction
- + RNNs for temporal summarisation of the extracted „features“

→ strong performance with respect to the number of parameter and training time
→ C-RNN better than CNN with comparable number of parameters

CNNs for Music Classification & Tagging

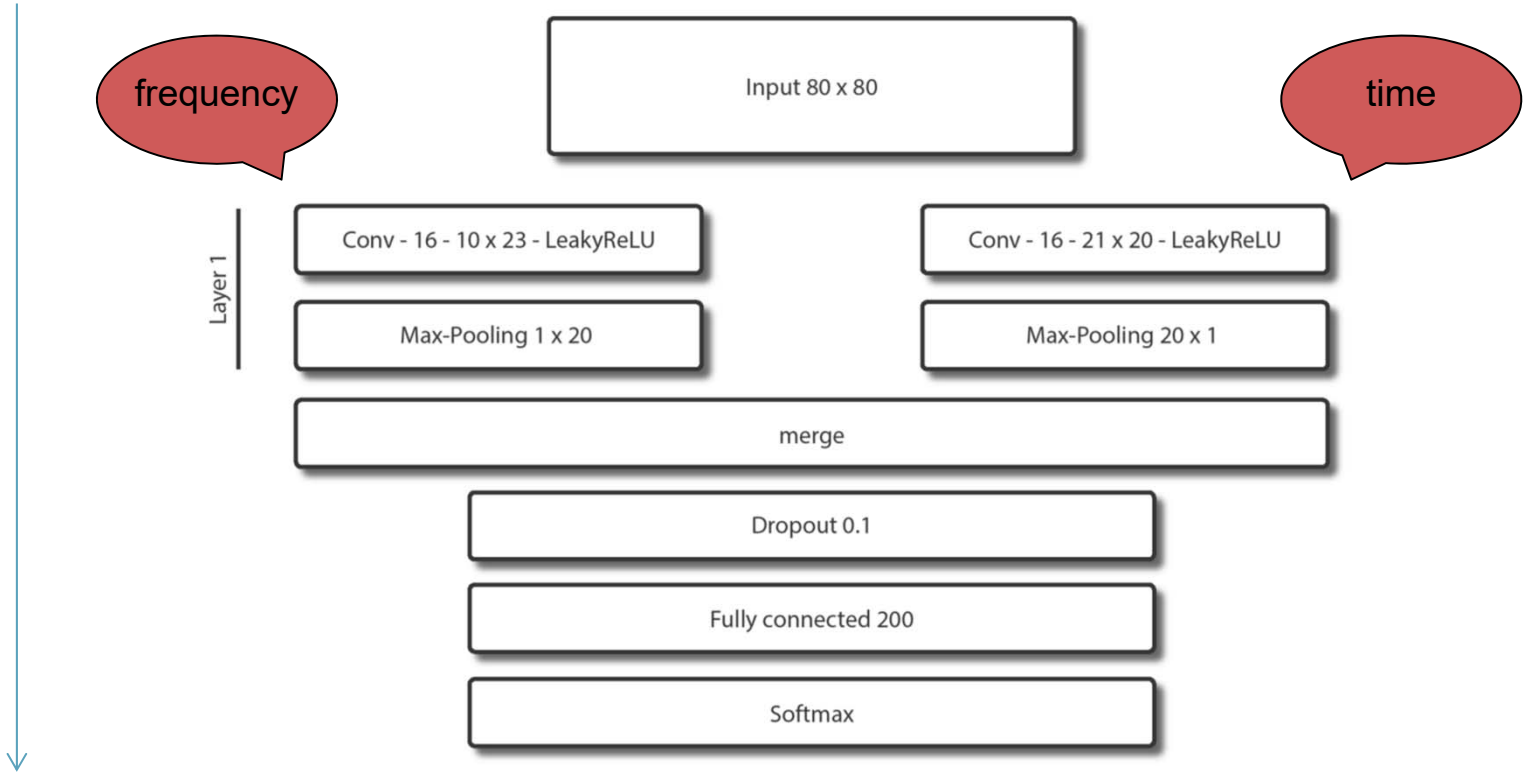


GTZAN	(Acc.):
MFCC	66%
(both joined)	78.1%
CNN-SVM	**89.8%**
state o.t.a.	94.5%

features are composed from
average pooling on weights from 5 layers

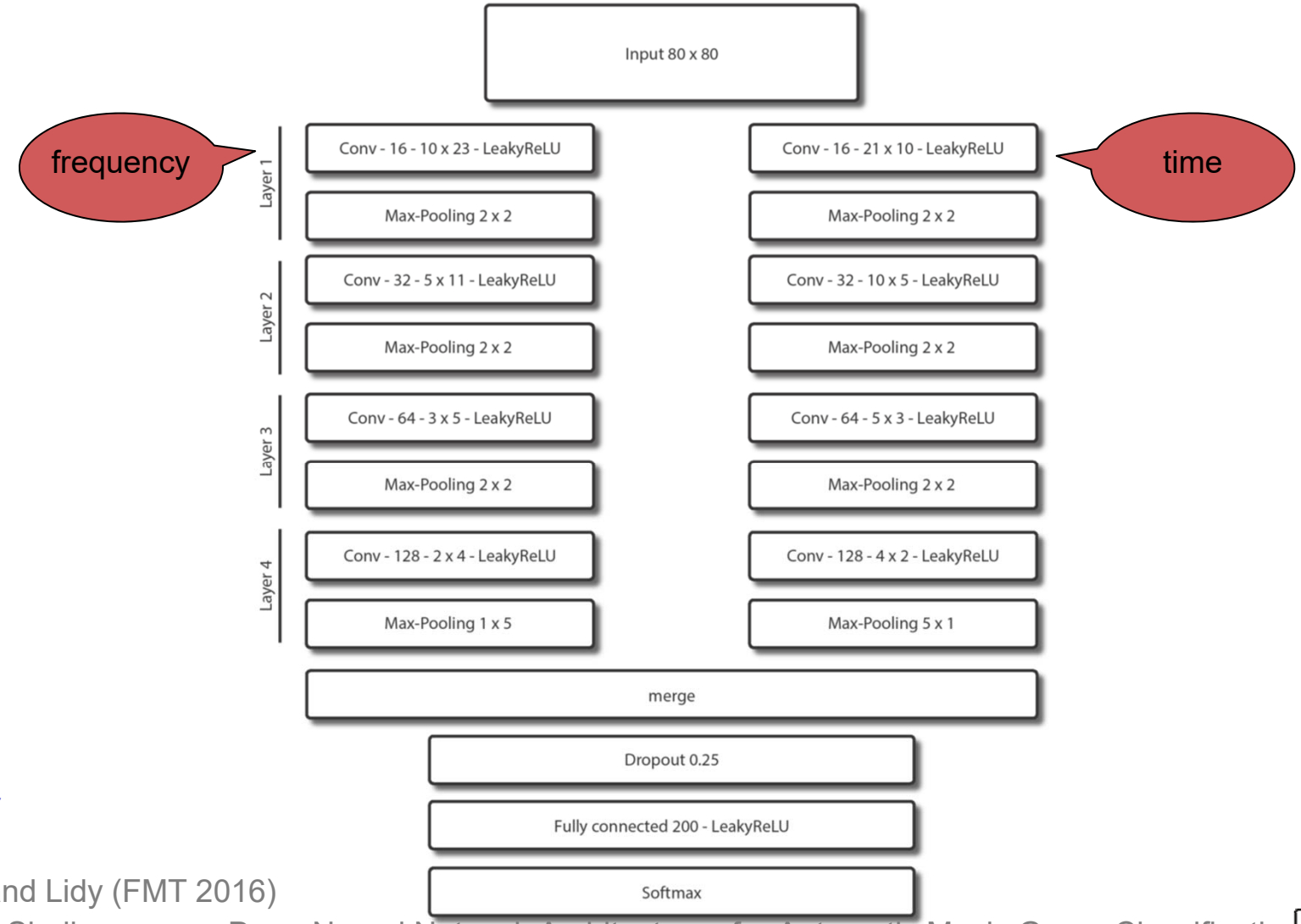
Keunwoo Choi
ISMIR 2017: Transfer learning for music classification and regression tasks
https://keunwoochoi.wordpress.com/2017/03/28/paper-is-out-transfer-learning-for-music-classification-and-regression-tasks-and-behind-the-scene-negative-results-etc/

Shallow vs. Deep: Shallow Architecture



Schindler and Lidy (FMT 2016)
Comparing Shallow versus Deep Neural Network Architectures for Automatic Music Genre Classification

Shallow vs. Deep: Deep Architecture



Schindler and Lidy (FMT 2016)
Comparing Shallow versus Deep Neural Network Architectures for Automatic Music Genre Classification

Results: Shallow vs. Deep

	100 epochs		200 epochs	
	Shallow	**Deep**	**Shallow**	**Deep**
GTZAN	78.1	**78.6**	**80.8**	80.6
ISMIRgenre	**85.5**	84.1	84.9	**85.1**
Latin	92.4	**94.4**	93.5	**95.1**
MSD	63.9	**67.2**	/	/

- Shallow Models: Similar or better performance on small datasets
- Deep Models: Advantage on bigger datasets

Accuracy by Max Probability
(paper contains other measures + variances)

Results: 100 vs. 200 epochs

	100 epochs		200 epochs	
	Shallow	**Deep**	**Shallow**	**Deep**
GTZAN	78.1	78.6	**80.8**	**80.6**
ISMIRgenre	**85.5**	84.1	84.9	**85.1**
Latin	92.4	94.4	**93.5**	**95.1**
MSD	63.9	67.2	/	/

- Models are improved by training 200 epochs (not if trained longer – overfitting)
- Except shallow model on ISMIRgenre

Accuracy by Max Probability
(paper contains other measures + variances)

Data Augmentation

Goal: Improve classifier by increasing invariance to irrelevant properties of input.

cat facing left

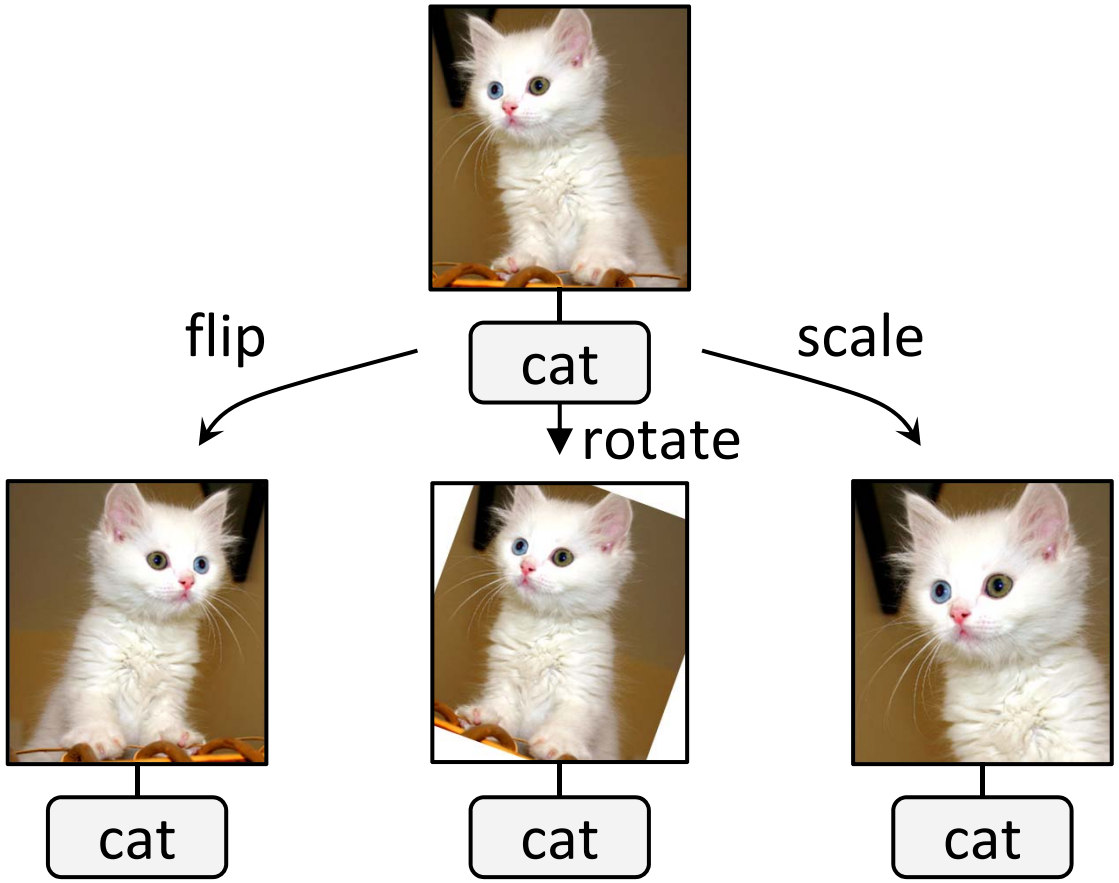
cat facing right

Possible solutions:

- Collect more training data
- Design invariant features or invariant model
- **Data augmentation**

Data Augmentation

Transform training data, let classifier learn to ignore irrelevant properties (e.g. orientation, rotation, scale, etc.)



Data Augmentation

Typical transformations:

- For images: horizontal flip, scale, rotation, color, contrast
- **For audio**: time stretching, pitch shifting, equalizer, …

Side effects:

- **Regularization**: Less likely to learn training data “by heart”
- **Increases** apparent **training set size**: Can train larger model
- Increases apparent training set size: Can use smaller dataset

Data Augmentation for Music

- Generate additional input data (from existing data set) to make the neural network generalize better
- **Time Stretching**: by factors 0.2x, 0.5x, 1.2x, 1.5x
- **Pitch Shifting**: –5, –2, +2, +5 semitones
- For each, 3 random segments from audio were taken (48 additional samples per audio file)

Enhancements & Modern Techniques

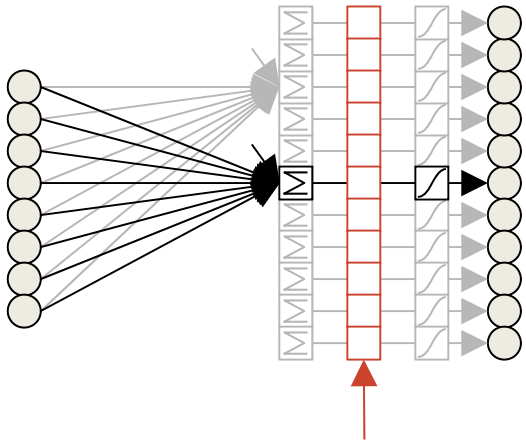
Batch Normalisation

Weight gradients depend on the unit activations.

Very small or very large activations lead to small/large gradients.

For input units, **standardizing** the **training data** avoids this.

Batch normalisation does the same for hidden units, even as mean/variance changes during training, computationally cheap.



batch normalisation

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

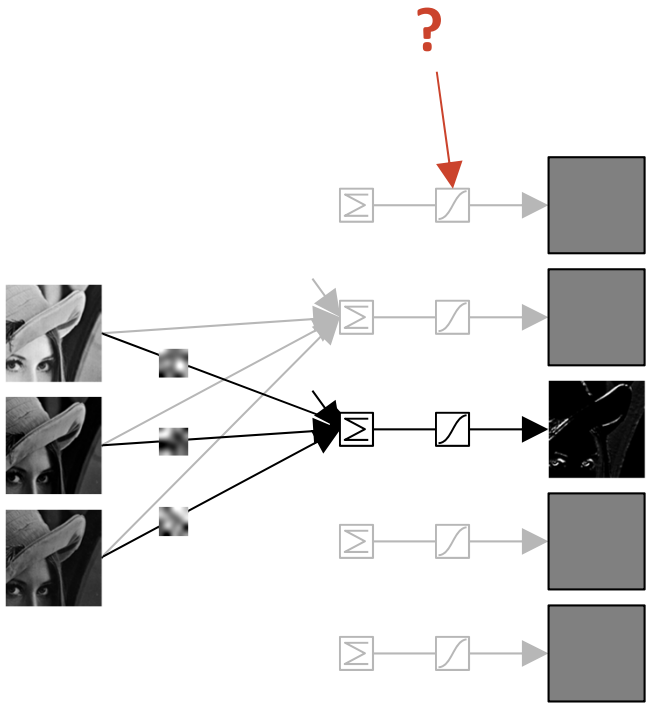
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

New Activation Functions

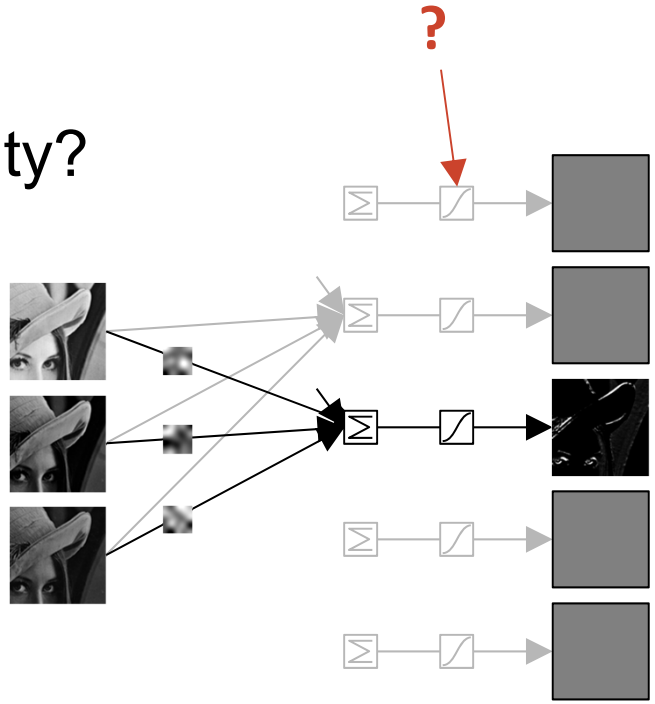
What shall the learnable function look like?



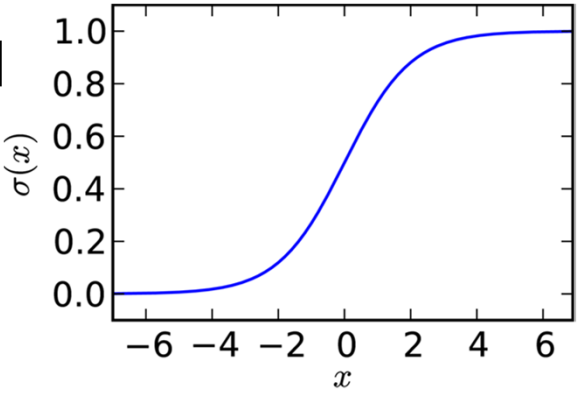
Linear Rectifier (ReLU)

What shall the learnable function look like?

- What *kind* of layers to use?
- Specifically, what kind of nonlinearity?



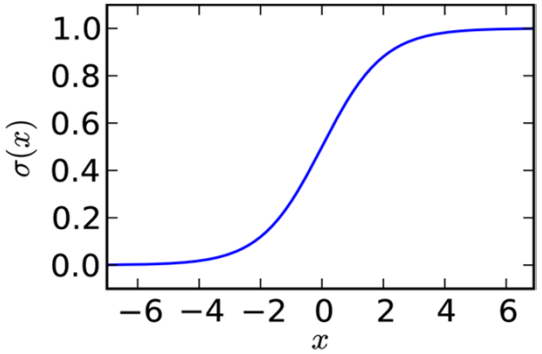
Sigmoid



Problem: saturates for large inputs (small slope, weak gradient!)

has nonzero mean (slows learning)

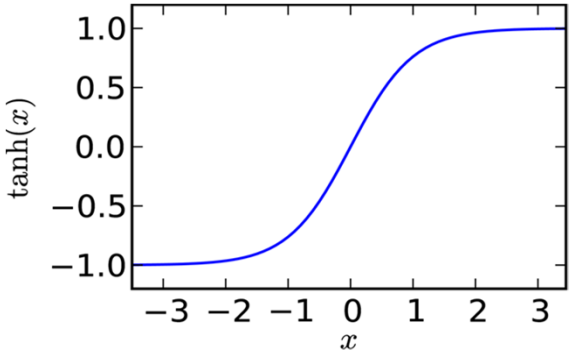
New Activation Functions:
Linear Rectifier (ReLU)



Sigmoid

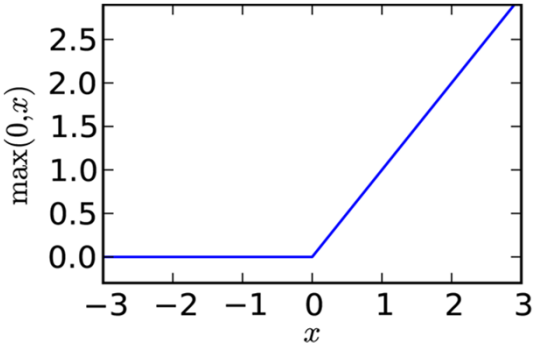
saturates for large inputs
(small slope, weak gradient!)

has nonzero mean
(slows learning)



TanH

saturates for large inputs
(small slope, weak gradient!)



ReLU

has nonzero mean
(slows learning)
has zero gradient for
negative input

Variants of ReLU: Leaky Rectifier (LReLU),
Parametric Rectifier (PReLU), ...

Benefits:
no saturation
low computational costs

ICML 2013: Rectifier Nonlinearities Improve Neural Network Acoustic Models

Even Newer Activation Functions

- ELU
- SeLU

Data Augmentation

Tutorial:

Deep Learning for Music Classification using Keras

Tutorial Agenda

III. Instrumental, Genre and Mood Analysis:

- Large-scale Music AI at Musimap
- Instrumental vs. Vocal Detection
- Genre Recognition
- Mood Detection
- Coding Examples

IV. Advanced Deep Learning:

- Similarity Retrieval
- Siamese Networks
- Learning Audio Representation from Tag Similarity
- Coding Examples