Deep Learning: Lecture 8

Alexander Schönhuth

UU

November 27, 2019

Recurrent Neural Networks

Recurrent Neural Networks

Introduction

- Unlike CNNs, which specialize in processing grid-/matrix-style input, *recurrent neural networks (RNNs)* specialize in processing sequences of values

$$\mathbf{x}^{(1)}, ..., \mathbf{x}^{(\tau)} \tag{1}$$

- *Advantages*:
  - RNNs can process very long sequences
  - RNNs can process sequences of flexible length
- To make this possible, they also employ *parameter sharing*
- *Additional literature*: "Supervised Sequence Labelling with Recurrent Neural Networks", A. Graves, 2012, https://www.springer.com/de/book/9783642247965

Recurrent Neural Networks

Introduction

- Unlike CNNs, which specialize in processing grid-/matrix-style input, *recurrent neural networks (RNNs)* specialize in processing sequences of values

$$\mathbf{x}^{(1)}, ..., \mathbf{x}^{(\tau)} \tag{1}$$

- *Advantages*:
  - RNNs can process very long sequences
  - RNNs can process sequences of flexible length
- To make this possible, they also employ *parameter sharing*
- *Additional literature*: "Supervised Sequence Labelling with Recurrent Neural Networks", A. Graves, 2012, https://www.springer.com/de/book/9783642247965

Recurrent Neural Networks

Introduction

- Unlike CNNs, which specialize in processing grid-/matrix-style input, *recurrent neural networks (RNNs)* specialize in processing sequences of values

$$\mathbf{x}^{(1)}, ..., \mathbf{x}^{(\tau)} \tag{1}$$

- *Advantages*:
  - RNNs can process very long sequences
  - RNNs can process sequences of flexible length
- To make this possible, they also employ *parameter sharing*
- *Additional literature*: "Supervised Sequence Labelling with Recurrent Neural Networks", A. Graves, 2012, https://www.springer.com/de/book/9783642247965

Recurrent Neural Networks

Introduction

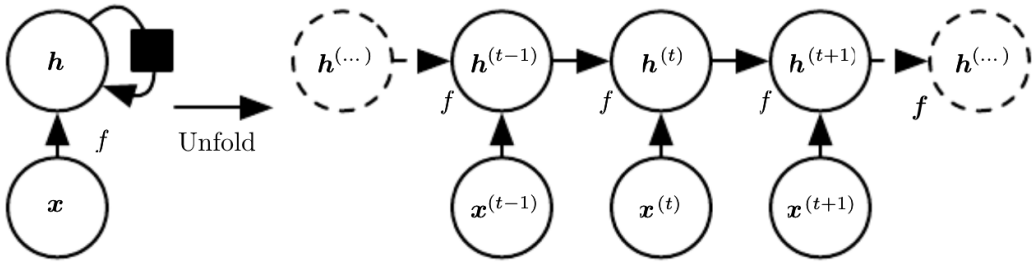
- Unlike CNNs, which specialize in processing grid-/matrix-style input, *recurrent neural networks (RNNs)* specialize in processing sequences of values

$$\mathbf{x}^{(1)}, ..., \mathbf{x}^{(\tau)} \tag{1}$$

- *Advantages*:
  - RNNs can process very long sequences
  - RNNs can process sequences of flexible length
- To make this possible, they also employ *parameter sharing*
- *Additional literature*: "Supervised Sequence Labelling with Recurrent Neural Networks", A. Graves, 2012, `https://www.springer.com/de/book/9783642247965`

Recurrent Neural Networks

Architecture



RNN with one hidden layer and no outputs

- Generating values:

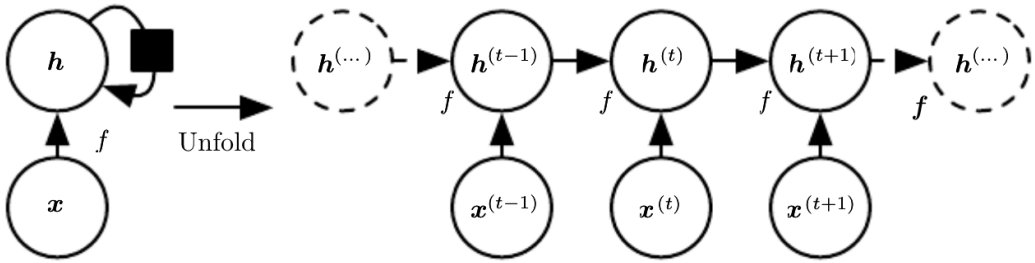
$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta) \tag{2}$$

- *Recurrence*:

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta) = f(f(\mathbf{h}^{(t-2)}, \mathbf{x}^{(t-1)}; \theta), \mathbf{x}^{(t)}; \theta) \tag{3}$$

Recurrent Neural Networks

Architecture



RNN with one hidden layer and no outputs

- Generating values:

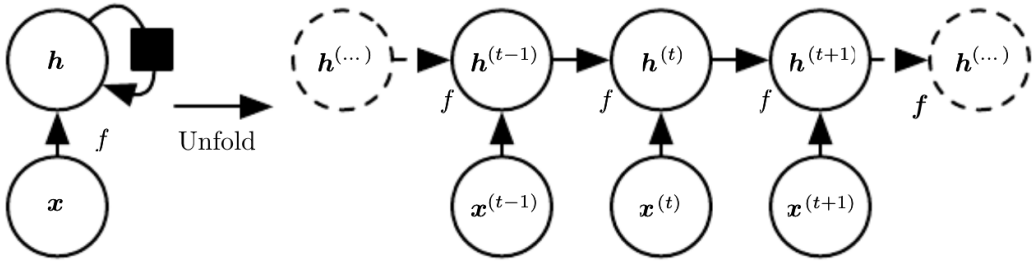
$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta) \tag{2}$$

- *Recurrence*:

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta) = f(f(\mathbf{h}^{(t-2)}, \mathbf{x}^{(t-1)}; \theta), \mathbf{x}^{(t)}; \theta) \tag{3}$$

Recurrent Neural Networks

Architecture



RNN with one hidden layer and no outputs

- Generating values:

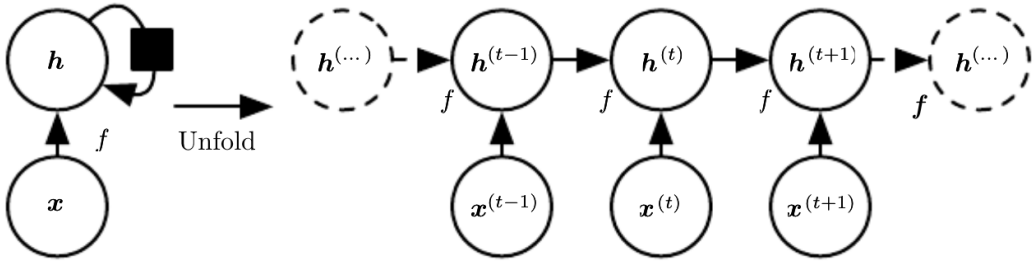
$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta) \tag{2}$$

- *Recurrence*:

$$\begin{aligned}\mathbf{h}^{(t)} &= f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta) = f(f(\mathbf{h}^{(t-2)}, \mathbf{x}^{(t-1)}; \theta), \mathbf{x}^{(t)}; \theta) \\ &= f(f(...(f(\mathbf{h}^{(0)}, \mathbf{x}^{(1)}; \theta), \mathbf{x}^{(2)}; \theta)...), \mathbf{x}^{(t)}; \theta)\end{aligned} \tag{3}$$

Recurrent Neural Networks

Architecture



RNN with one hidden layer and no outputs

- Generating values:

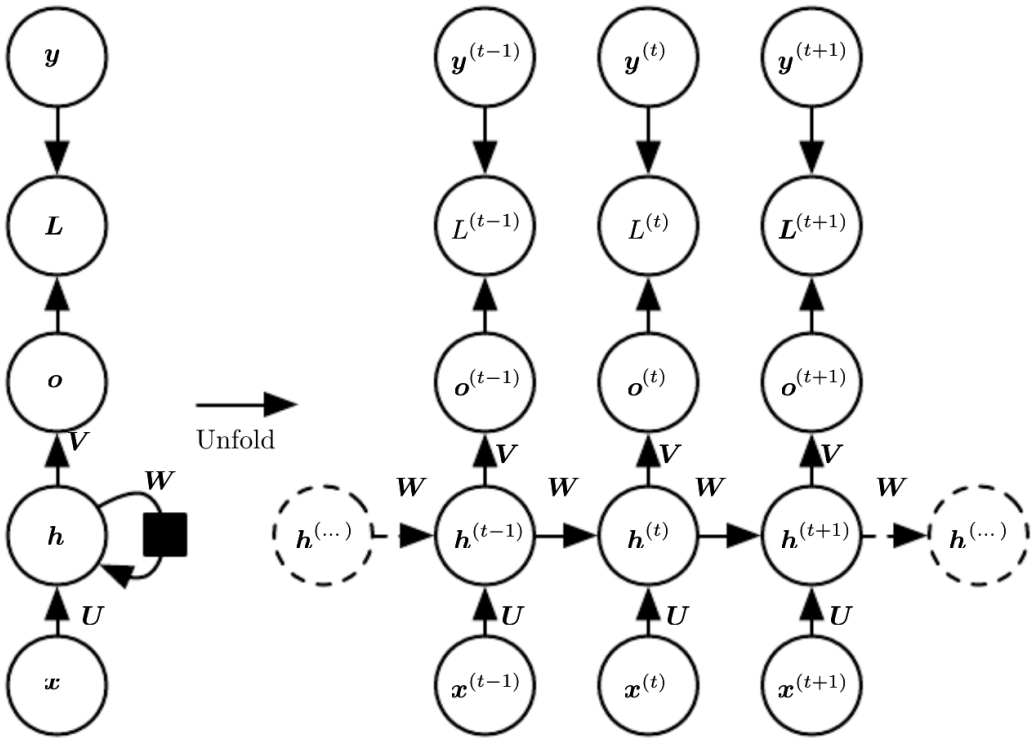
$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta) \tag{2}$$

- *Recurrence*:

$$\begin{aligned} \mathbf{h}^{(t)} &= f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta) = f(f(\mathbf{h}^{(t-2)}, \mathbf{x}^{(t-1)}; \theta), \mathbf{x}^{(t)}; \theta) \\ &= f(f(...(f(\mathbf{h}^{(0)}, \mathbf{x}^{(1)}; \theta), \mathbf{x}^{(2)}; \theta)...), \mathbf{x}^{(t)}; \theta) \\ &=: g^{(t)}(\mathbf{x}^{(t)}, \mathbf{x}^{(t-1)}, ..., \mathbf{x}^{(1)}; \theta) \end{aligned} \tag{3}$$

Recurrent Neural Networks

Architecture



Output at each time step, recurrent connections between hidden units

Recurrent Neural Networks

Forward Propagation

Let σ be a suitable activation function. Then forward propagation in RNN's of the type from the slide before proceeds as follows:

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \tag{4}$$

$$\mathbf{h}^{(t)} = \sigma(\mathbf{a}^{(t)}) \tag{5}$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)} \tag{6}$$

$$\hat{\mathbf{y}}^{(t)} = \text{softmax}(\mathbf{o}^{(t)}) \tag{7}$$

where $\mathbf{b}, \mathbf{c}$ are the bias vectors along with $\mathbf{W}, \mathbf{U}$ and $\mathbf{V}$, respectively.

Recurrent Neural Networks

Computing Cost

- Let $\mathbf{y} = (y^{(1)}, ..., y^{(t)})$ be true labels for the sequence $\mathbf{x}^{(1)}, ..., \mathbf{x}^{(t)}$.
- Then we compute the cost C as

$$C(\{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(\tau)}\}, \{y^{(1)}, ..., y^{(\tau)}\}) = \sum_{t=1}^{\tau} C^{(t)} \tag{8}$$

where

$$C^{(t)} = -\log p_{\text{model}}(y^{(t)} \mid \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(t)}\}) \tag{9}$$

and p_{model} refers to the probability computed by application of *softmax* to $\mathbf{o}^{(t)}$.

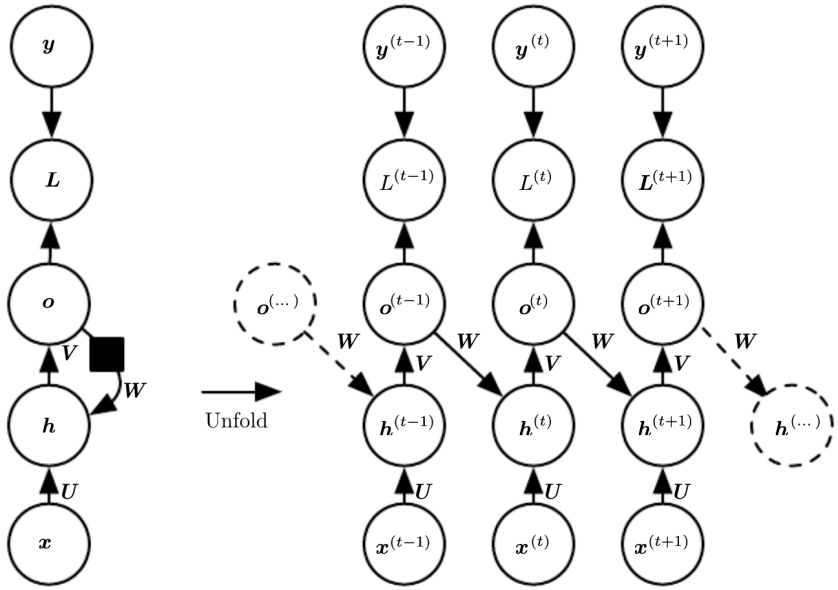
Recurrent Neural Networks

Computing Gradients

- Computing gradients does not involve any particular complications.
- See `http://www.deeplearningbook.org/contents/rnn.html`, 10.2.2 (☞ *Homework*)

Recurrent Neural Networks

Architecture II



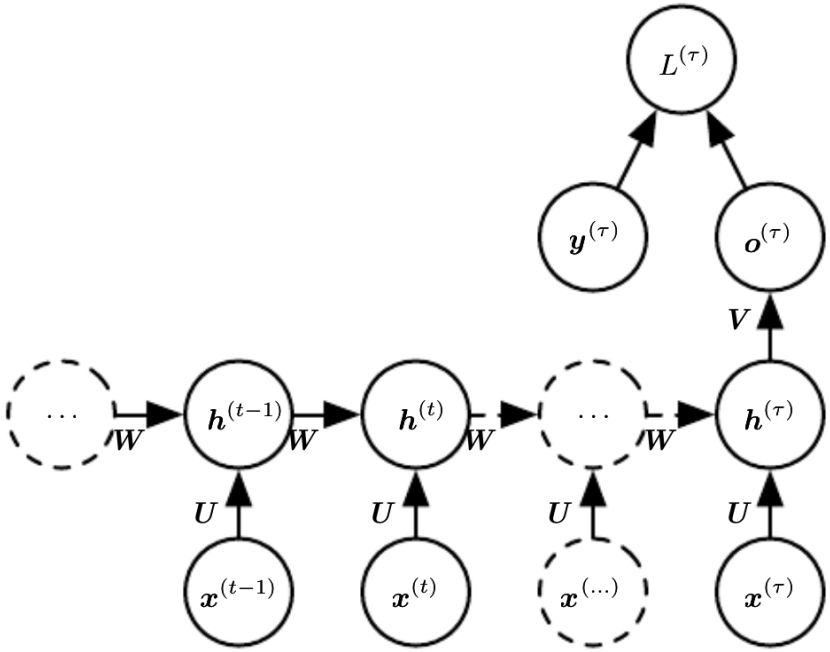
RNN where output units connect to hidden units:

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{o}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \tag{10}$$

- Less powerful, but easier to train

Recurrent Neural Networks

Architecture II

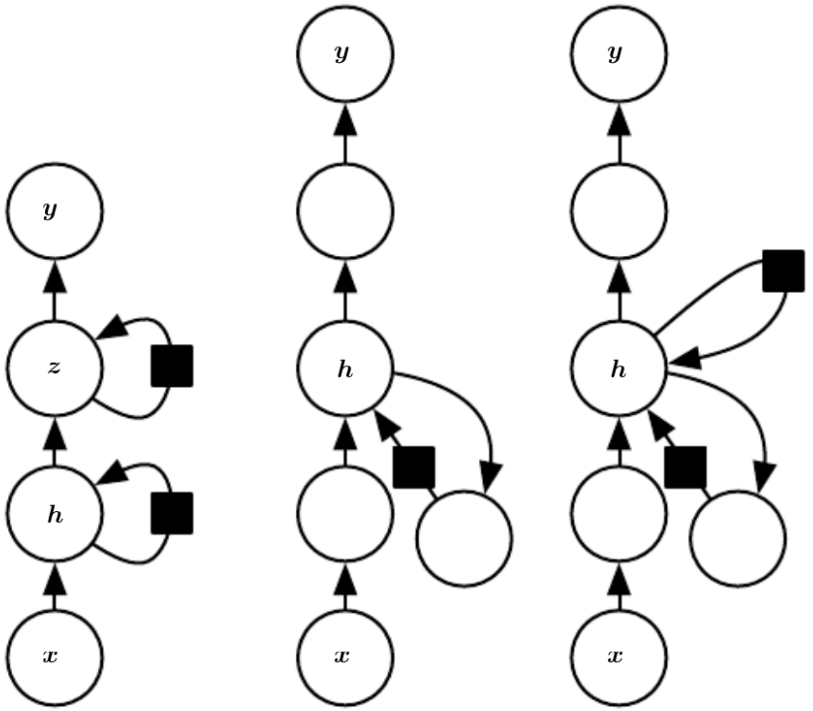


RNN that generates one, summarizing output

- Can be used for generation of fixed-size representation used as input for further processing

Recurrent Neural Networks

Going Deep



(a) (b) (c)

(a) Extra layer of hidden units, all operating the same way

(b) Introduction of units between hidden units

(c) Like (b), but with skip connections

Recurrent Neural Networks

Motivation / Summary

- *Time Dynamics*: Model that behaviour of a network may vary over time
- *Recurrence*: Model that output derived from input may depend on inputs seen earlier
- Particularly useful for analyzing data / processes that change over time
  - Speech recognition
  - Natural language processing
- NNs have trouble solving certain problems conventional approaches are good at and vice versa
- *Recurrent Neural Networks* are an attempt to have a unifying model that is good at everything

Recurrent Neural Networks

Motivation / Summary

- *Time Dynamics*: Model that behaviour of a network may vary over time
- *Recurrence*: Model that output derived from input may depend on inputs seen earlier
- Particularly useful for analyzing data / processes that change over time
  - Speech recognition
  - Natural language processing
- NNs have trouble solving certain problems conventional approaches are good at and vice versa
- *Recurrent Neural Networks* are an attempt to have a unifying model that is good at everything

Recurrent Neural Networks

Motivation / Summary

- *Time Dynamics*: Model that behaviour of a network may vary over time
- *Recurrence*: Model that output derived from input may depend on inputs seen earlier
- Particularly useful for analyzing data / processes that change over time
  - Speech recognition
  - Natural language processing
- NNs have trouble solving certain problems conventional approaches are good at and vice versa
- *Recurrent Neural Networks* are an attempt to have a unifying model that is good at everything

Recurrent Neural Networks

Motivation / Summary

- *Time Dynamics*: Model that behaviour of a network may vary over time
- *Recurrence*: Model that output derived from input may depend on inputs seen earlier
- Particularly useful for analyzing data / processes that change over time
  - Speech recognition
  - Natural language processing
- NNs have trouble solving certain problems conventional approaches are good at and vice versa
- *Recurrent Neural Networks* are an attempt to have a unifying model that is good at everything

Recurrent Neural Networks

Further Models

- *Bidirectional RNNs*
  - For computing $\mathbf{h}^{(s)}$ take both earlier ($t = 1, \ldots, s-1$) and later ($t = s+1, \ldots, \tau$) values into account
  - Successful in handwriting and speech recognition
- *Encoder-Decoder Sequence to Sequence Architectures*
  - Ordinary RNNs map sequences to sequences of same length
  - Encoder-Decoder RNNs map sequences to sequences of not necessarily the same length
  - Applications: Translations, question answering
- See `http://www.deeplearningbook.org/contents/rnn.html`, 10.3, 10.4

Recurrent Neural Networks

Further Models

- *Bidirectional RNNs*
  - For computing $\mathbf{h}^{(s)}$ take both earlier ($t = 1, \ldots, s-1$) and later ($t = s+1, \ldots, \tau$) values into account
  - Successful in handwriting and speech recognition
- *Encoder-Decoder Sequence to Sequence Architectures*
  - Ordinary RNNs map sequences to sequences of same length
  - Encoder-Decoder RNNs map sequences to sequences of not necessarily the same length
  - Applications: Translations, question answering
- See `http://www.deeplearningbook.org/contents/rnn.html`, 10.3, 10.4

Recurrent Neural Networks

Further Models

- *Bidirectional RNNs*
  - For computing $\mathbf{h}^{(s)}$ take both earlier ($t = 1, ..., s-1$) and later ($t = s+1, ..., \tau$) values into account
  - Successful in handwriting and speech recognition
- *Encoder-Decoder Sequence to Sequence Architectures*
  - Ordinary RNNs map sequences to sequences of same length
  - Encoder-Decoder RNNs map sequences to sequences of not necessarily the same length
  - Applications: Translations, question answering
- See `http://www.deeplearningbook.org/contents/rnn.html`, 10.3, 10.4

Recurrent Neural Networks

Further Models

- *Recursive Neural Networks*
  - Generalize RNNs from sequence-style to tree-shaped input
  - Inputs are transformed into outputs according to a hierarchical structure
  - *Applications*: Process data structures as input to NNs; language processing; computer vision
- *Long Short-Term Memory RNNs*
  - *Motivation*: In RNNs, gradients vanish or explode easily, so integration of long-term dependencies remains difficult
  - LSTM RNNs present a (tricky) technique to avoid this
  - LSTM RNNs are currently the most successful RNN model
- See `http://www.deeplearningbook.org/contents/rnn.html`, 10.6, 10.10

Recurrent Neural Networks

Further Models

- *Recursive Neural Networks*
  - Generalize RNNs from sequence-style to tree-shaped input
  - Inputs are transformed into outputs according to a hierarchical structure
  - *Applications*: Process data structures as input to NNs; language processing; computer vision
- *Long Short-Term Memory RNNs*
  - *Motivation*: In RNNs, gradients vanish or explode easily, so integration of long-term dependencies remains difficult
  - LSTM RNNs present a (tricky) technique to avoid this
  - LSTM RNNs are currently the most successful RNN model
- See `http://www.deeplearningbook.org/contents/rnn.html`, 10.6, 10.10

Recurrent Neural Networks

Further Models

- *Recursive Neural Networks*
  - Generalize RNNs from sequence-style to tree-shaped input
  - Inputs are transformed into outputs according to a hierarchical structure
  - *Applications*: Process data structures as input to NNs; language processing; computer vision
- *Long Short-Term Memory RNNs*
  - *Motivation*: In RNNs, gradients vanish or explode easily, so integration of long-term dependencies remains difficult
  - LSTM RNNs present a (tricky) technique to avoid this
  - LSTM RNNs are currently the most successful RNN model
- See `http://www.deeplearningbook.org/contents/rnn.html`, 10.6, 10.10

Going Deeper – Architecture Development

Motivation: Reminder

The Universal Approximation Theorem

- *Accuracy of approximation of (arbitrary) function f by NN $\hat{f}$ increases exponentially on increasing number of hidden layers*
- See [Cybenko, 1989, doi:10.1007/BF02551274], [Hornik, 1991, doi:10.1016/0893-6080(91)90009-T]
- See [Montufar et al., 2014]:

  `https://arxiv.org/pdf/1402.1869.pdf`
- Explanation:

  `http://neuralnetworksanddeeplearning.com/chap4.html`
- Further resources for the following:
  - `https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html`
  - `https://www.analyticsvidhya.com/blog/2017/08/10-advanced-deep-learning-architectures-data-scientists/`

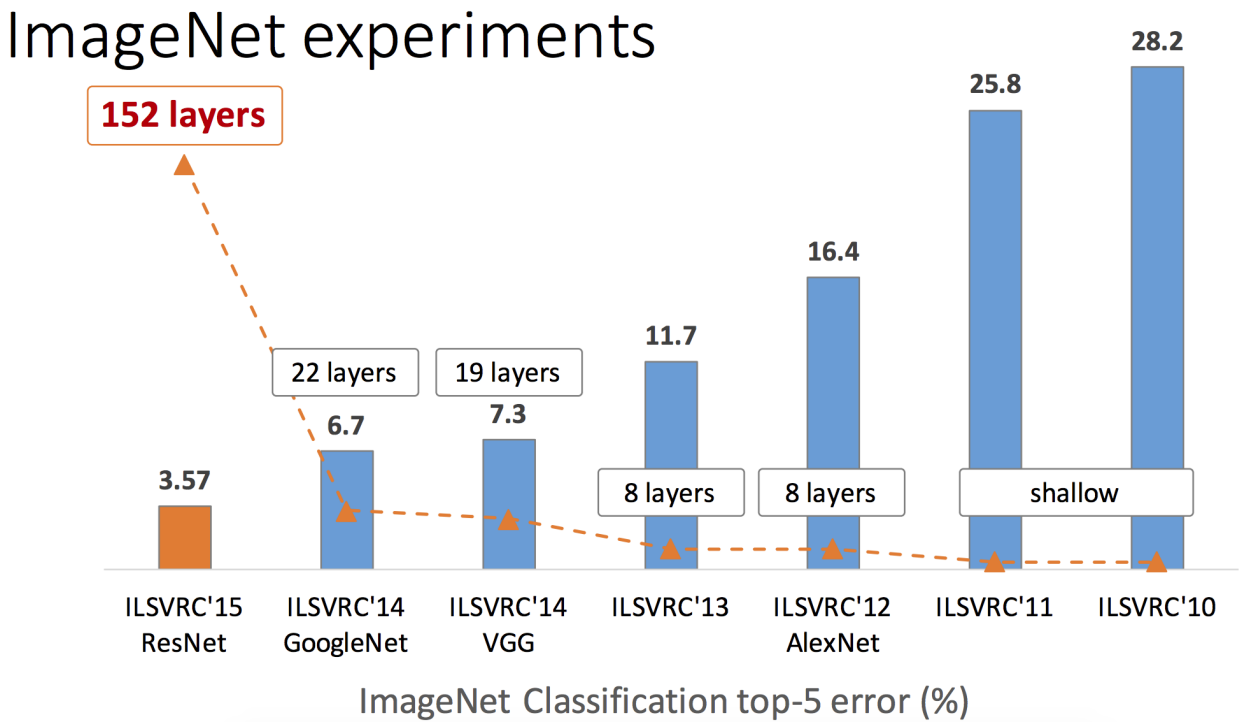
ImageNet and ILSVRC

Dataset and First Results

ImageNet examples: “beading plane”, “brown root rot fungus”, “scalded milk”, “common roundworm”

- *ImageNet dataset*: 16 million full color images; 20 000 categories
- *Starting point*: Le, Ranzato, Monga, Devin, Chen, Corrado, Dean & Ng: “Building high-level features using large scale unsupervised learning”, 2012, `https://ai.google/research/pubs/pub38115` achieved 15.3 % test accuracy
- *ILSVRC*: Image-Net Large-Scale Visual Recognition Challenge
  - *2012*: 1000 categories; Training 1.2 million images; Validation 50 000 images; Test 150 000 images

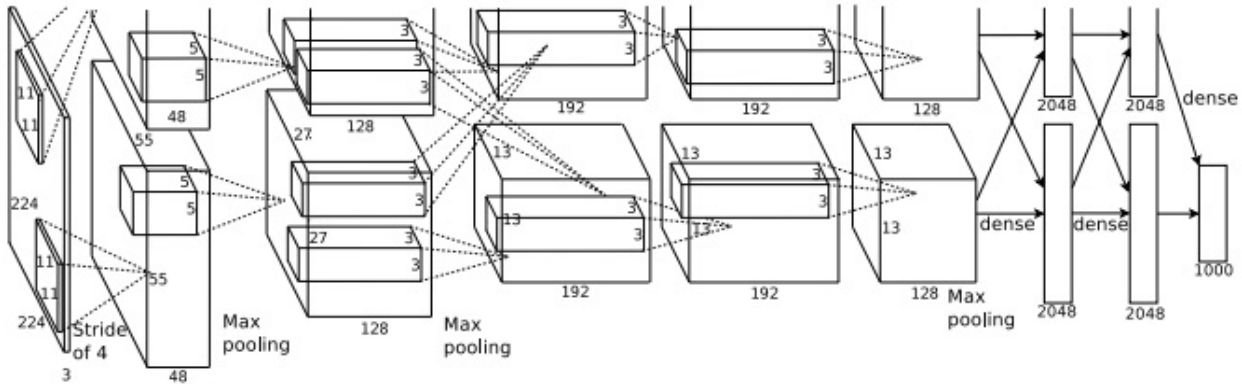
Going Deeper



https://icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf; Note: correct error rate for AlexNet is 15.4%

AlexNet

AlexNet

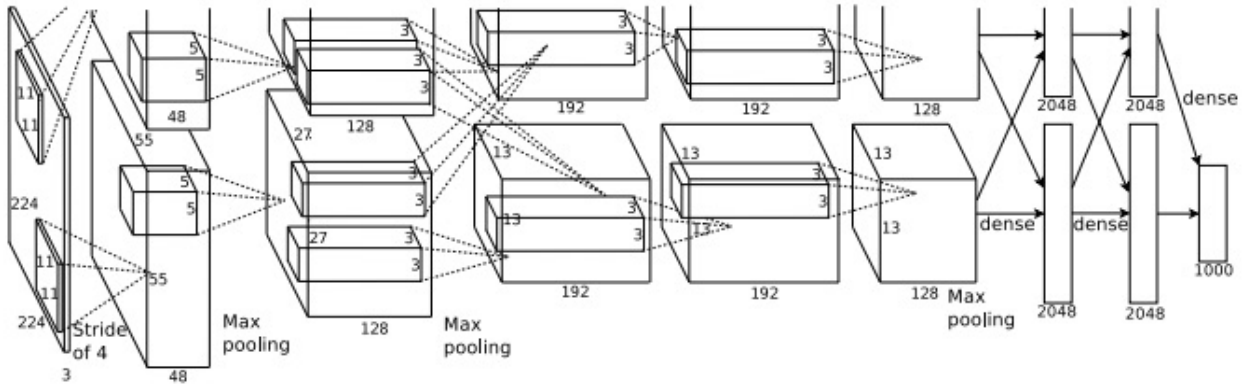


AlexNet from Krizhevsky, Sutskever & Hinton, 2012,

`http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

- *Input layer*: $3 \times 224 \times 224$ (= 224×224 RGB pixels)
- *First hidden layer*: convolutional + max-pooling
  - 11×11 sized filters
  - stride length 4
  - 96 feature maps in total
  - 48 each are run on separate GPU;
  - max-pooling [also in later layers] is 3×3, 2 pixels apart (so overlap)

AlexNet

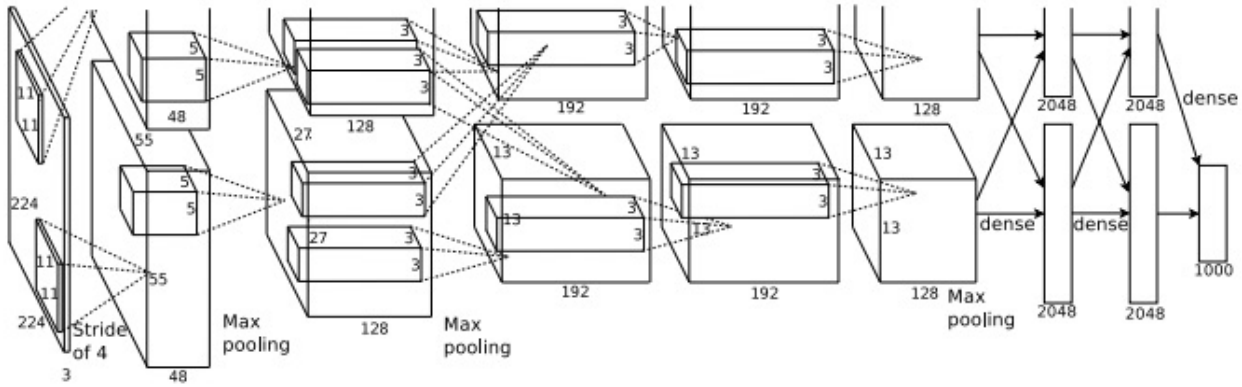


AlexNet from Krizhevsky, Sutskever & Hinton, 2012,

`http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

- *Input layer*: $3 \times 224 \times 224$ (= 224×224 RGB pixels)
- *First hidden layer*: convolutional + max-pooling
  - 11×11 sized filters
  - stride length 4
  - 96 feature maps in total
  - 48 each are run on separate GPU;
  - max-pooling [also in later layers] is 3×3, 2 pixels apart (so overlap)

AlexNet

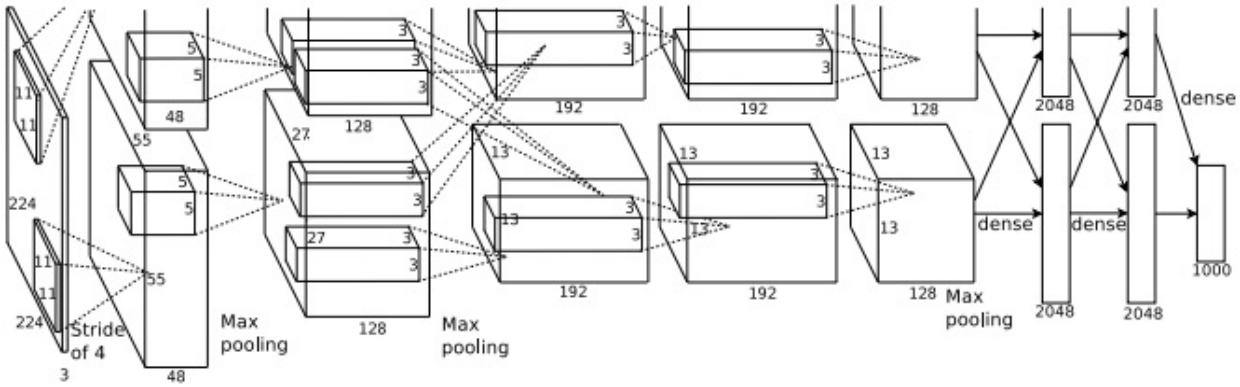


AlexNet from Krizhevsky, Sutskever & Hinton, 2012,

`http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

- *Input layer*: $3 \times 224 \times 224$ (= 224×224 RGB pixels)
- *First hidden layer*: convolutional + max-pooling
  - 11×11 sized filters
  - stride length 4
  - 96 feature maps in total
  - 48 each are run on separate GPU;
  - max-pooling [also in later layers] is 3×3, 2 pixels apart (so overlap)

AlexNet

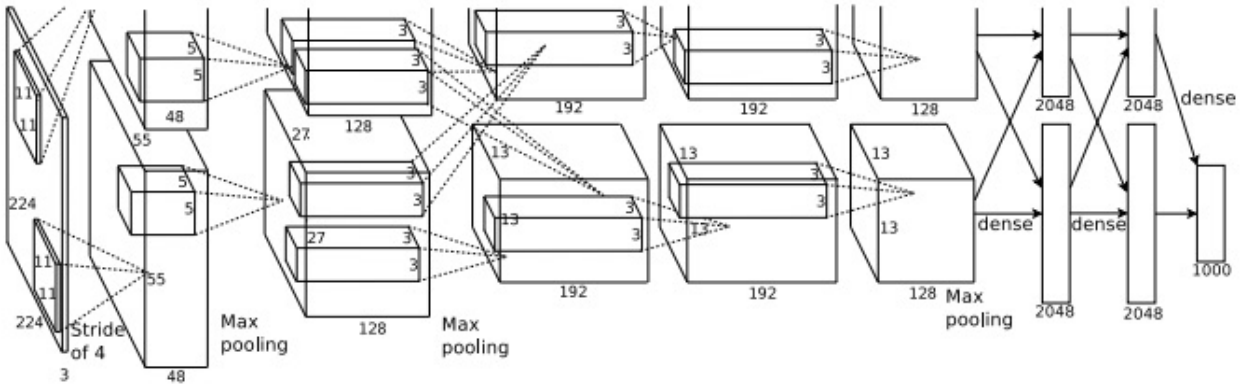


AlexNet from Krizhevsky, Sutskever & Hinton, 2012,

`http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

- *Input layer*: $3 \times 224 \times 224$ (= 224×224 RGB pixels)
- *First hidden layer*: convolutional + max-pooling
  - 11×11 sized filters
  - stride length 4
  - 96 feature maps in total
  - 48 each are run on separate GPU;
  - max-pooling [also in later layers] is 3×3, 2 pixels apart (so overlap)

AlexNet

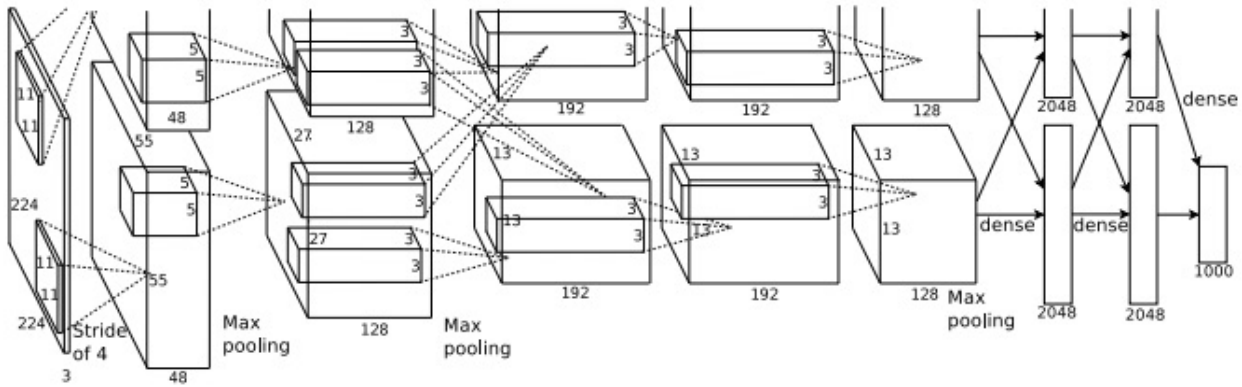


AlexNet from Krizhevsky, Sutskever & Hinton, 2012,

`http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

- *Input layer*: $3 \times 224 \times 224$ (= 224×224 RGB pixels)
- *First hidden layer*: convolutional + max-pooling
  - 11×11 sized filters
  - stride length 4
  - 96 feature maps in total
  - 48 each are run on separate GPU;
  - max-pooling [also in later layers] is 3×3, 2 pixels apart (so overlap)

AlexNet

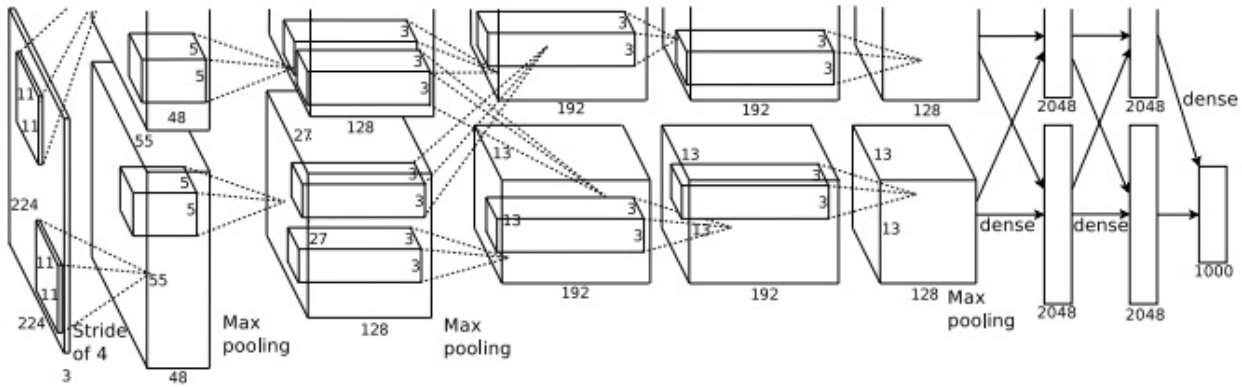


AlexNet from Krizhevsky, Sutskever & Hinton, 2012,

`http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

- *Second hidden layer*: convolutional + max-pooling
  - 5×5 sized filters
  - 256 feature maps in total
  - 128 each are run on separate GPU; each of those receive only 48 input channels

AlexNet

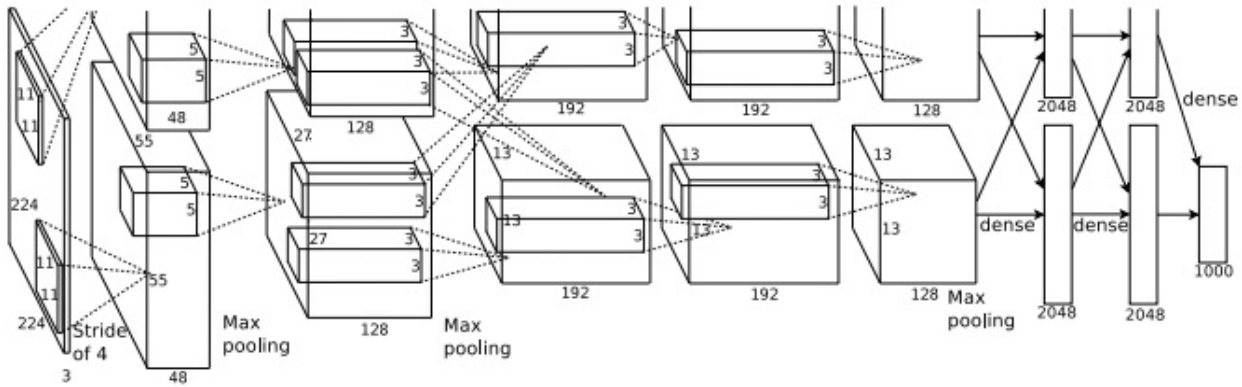


AlexNet from Krizhevsky, Sutskever & Hinton, 2012,

`http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

- *Second hidden layer*: convolutional + max-pooling
  - 5×5 sized filters
  - 256 feature maps in total
  - 128 each are run on separate GPU; each of those receive only 48 input channels

AlexNet

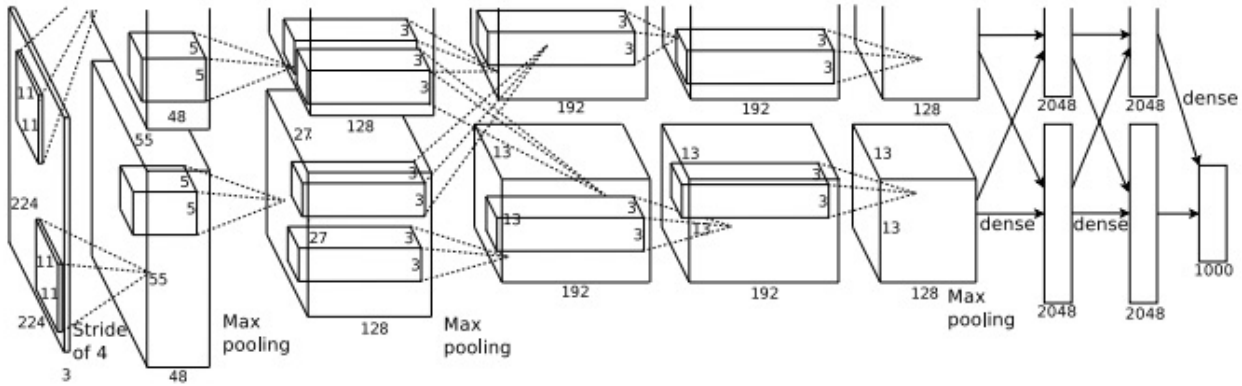


AlexNet from Krizhevsky, Sutskever & Hinton, 2012,

`http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

- *Second hidden layer*: convolutional + max-pooling
  - 5×5 sized filters
  - 256 feature maps in total
  - 128 each are run on separate GPU; each of those receive only 48 input channels

AlexNet

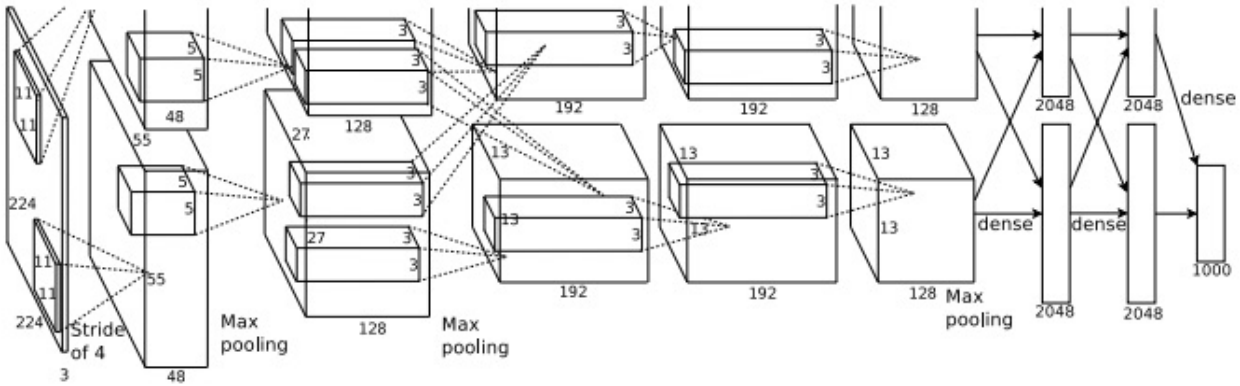


AlexNet from Krizhevsky, Sutskever & Hinton, 2012,

`http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

- *Third, fourth, fifth layer*: convolutional (no max-pooling)
  - *Third*: 384 feature maps, 3×3, and 256 input channels [with some inter-GPU communication]
  - *Fourth*: [384, 3×3, 192]; *Fifth*: [256, 3×3, 192]
- *Sixth, seventh layer*: fully connected, 4 096 neurons
- *Output layer*: 1000-unit softmax layer

AlexNet

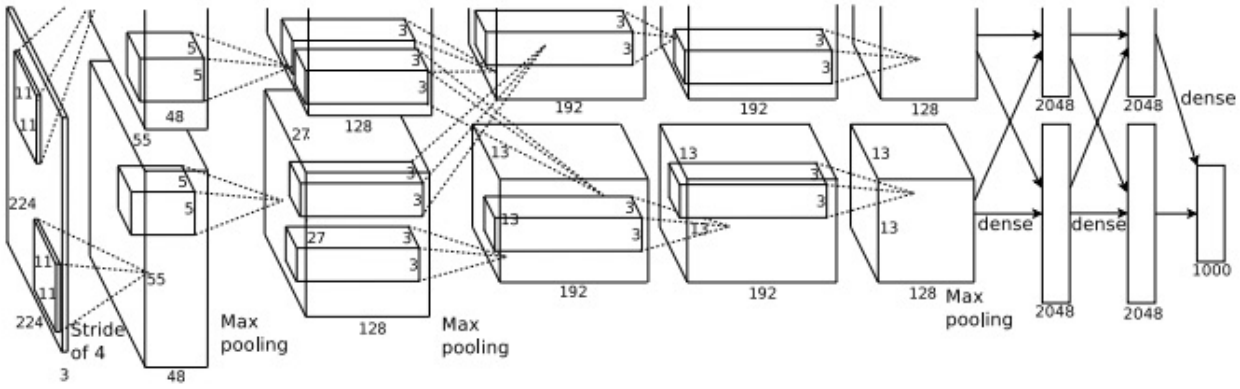


AlexNet from Krizhevsky, Sutskever & Hinton, 2012,

`http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

- *Third, fourth, fifth layer*: convolutional (no max-pooling)
  - *Third*: 384 feature maps, 3×3, and 256 input channels [with some inter-GPU communication]
  - *Fourth*: [384, 3×3, 192]; *Fifth*: [256, 3×3, 192]
- *Sixth, seventh layer*: fully connected, 4 096 neurons
- *Output layer*: 1000-unit softmax layer

AlexNet

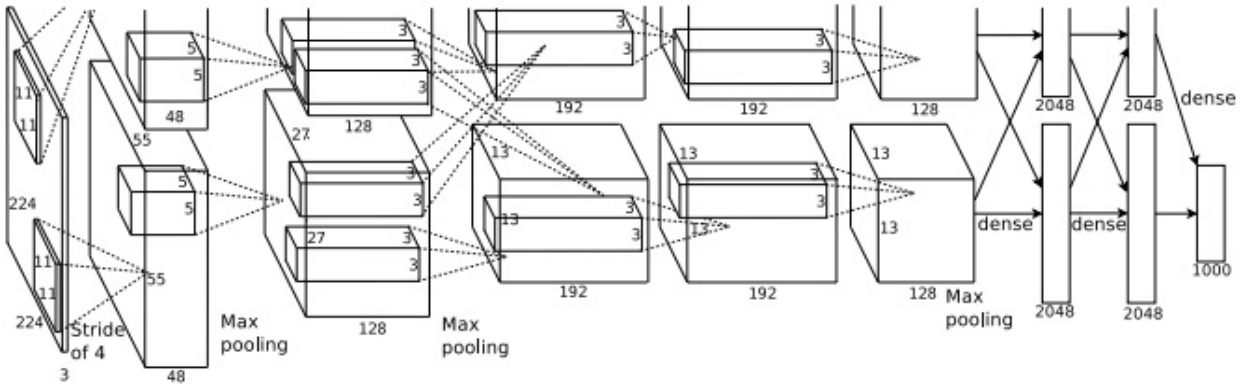


AlexNet from Krizhevsky, Sutskever & Hinton, 2012,

`http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

- *Third, fourth, fifth layer*: convolutional (no max-pooling)
  - *Third*: 384 feature maps, 3×3, and 256 input channels [with some inter-GPU communication]
  - *Fourth*: [384, 3×3, 192]; *Fifth*: [256, 3×3, 192]
- *Sixth, seventh layer*: fully connected, 4 096 neurons
- *Output layer*: 1000-unit softmax layer

AlexNet



AlexNet from Krizhevsky, Sutskever & Hinton, 2012,

`http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

- *Third, fourth, fifth layer*: convolutional (no max-pooling)
  - *Third*: 384 feature maps, 3×3, and 256 input channels [with some inter-GPU communication]
  - *Fourth*: [384, 3×3, 192]; *Fifth*: [256, 3×3, 192]
- *Sixth, seventh layer*: fully connected, 4 096 neurons
- *Output layer*: 1000-unit softmax layer

AlexNet

Further Features and ILSVRC performance

- Further architecture features:
  - Rectified linear neurons
  - *Regularization*: L2 + dropout
  - *Optimization*: Momentum-based stochastic gradient descent
- *ILSVRC Performance*: Achieved 63.3 % accuracy; 84.7 % "top-5 accuracy" (if the correct label is among the top-5 predictions of the NN), followed by 73.8 % by the second-best performing contestant
- *Why important?* AlexNet's amazing performance rate meant the "coming out" for CNNs in computer vision

AlexNet

Further Features and ILSVRC performance

- Further architecture features:
  - Rectified linear neurons
  - *Regularization*: L2 + dropout
  - *Optimization*: Momentum-based stochastic gradient descent
- *ILSVRC Performance*: Achieved 63.3 % accuracy; 84.7 % "top-5 accuracy" (if the correct label is among the top-5 predictions of the NN), followed by 73.8 % by the second-best performing contestant
- *Why important?* AlexNet's amazing performance rate meant the "coming out" for CNNs in computer vision

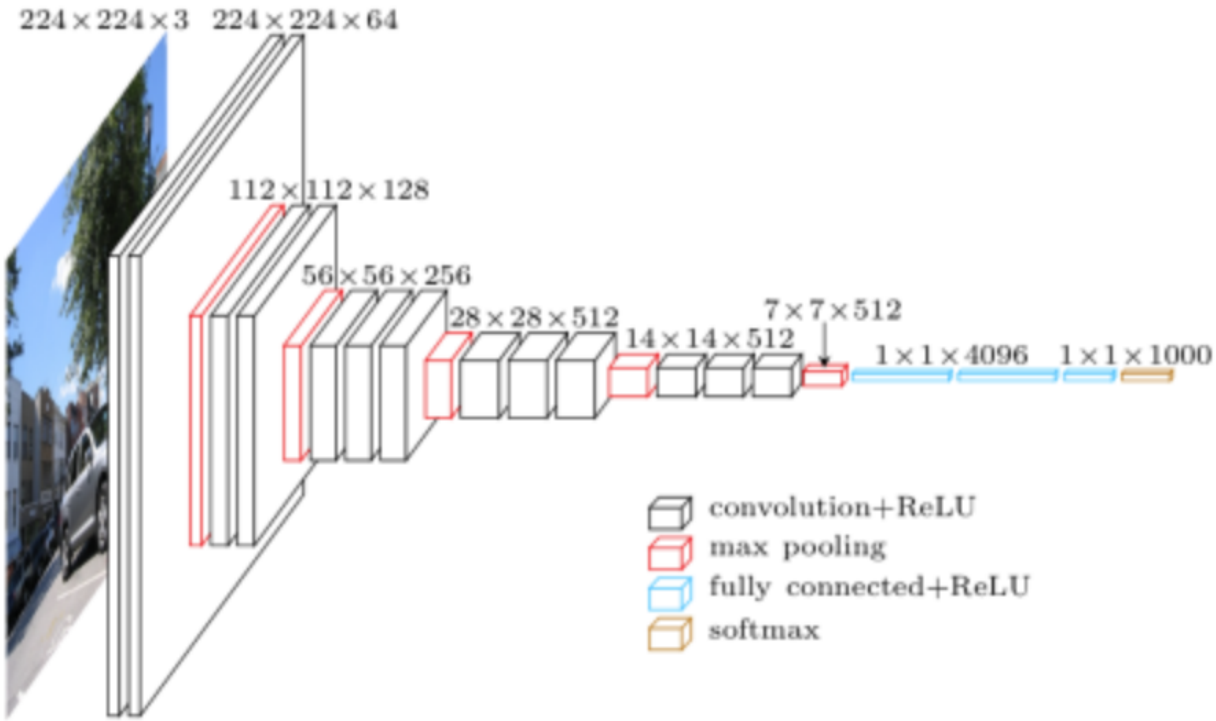
AlexNet

Further Features and ILSVRC performance

- Further architecture features:
  - Rectified linear neurons
  - *Regularization*: L2 + dropout
  - *Optimization*: Momentum-based stochastic gradient descent
- *ILSVRC Performance*: Achieved 63.3 % accuracy; 84.7 % "top-5 accuracy" (if the correct label is among the top-5 predictions of the NN), followed by 73.8 % by the second-best performing contestant
- *Why important?* AlexNet's amazing performance rate meant the "coming out" for CNNs in computer vision

VGG

VGG



VGG from Simonyan & Zisserman, 2014

`https://arxiv.org/pdf/1409.1556v6.pdf`

VGG

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB imag)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Different architectures tested in Simonyan & Zisserman, 2014

The yellow architecture yielded best performance in ILSVRC 2014

VGG

Features

- Uses only 3×3 filters;
  - argues that consecutive application of 3×3 filters yields virtual larger filters
  - For example, two 3×3 in a row yield virtual 5×5 filter
- 19(!) layers
- Less parameters than AlexNet
- ReLU layers and batch gradient descent

VGG

Features

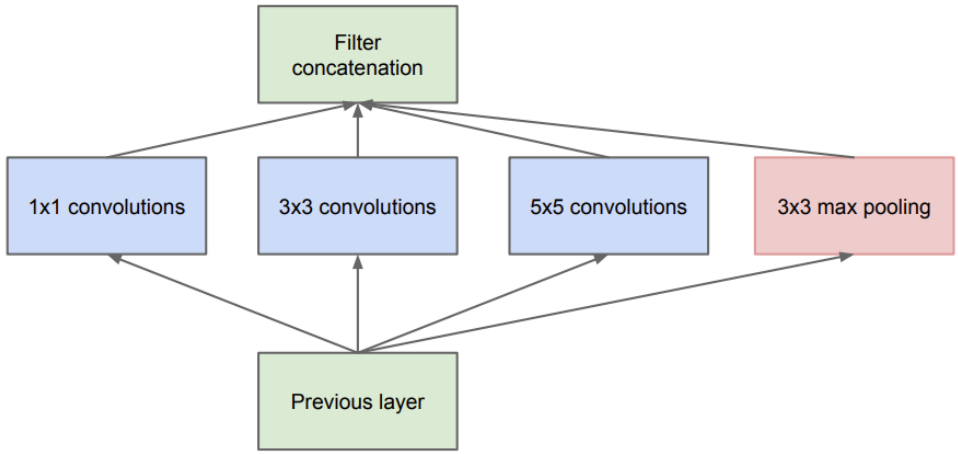
- *Advantages*:
  - Top-5 accuracy: 92.7% (vs. 83.7% from AlexNet)
  - Good architecture for benchmarking
  - Pre-trained networks available
- *Disadvantage*: very slow to train, training needs two to three weeks
- *Why important*:
  - Reinforced that CNNs have to be sufficiently deep to work well
  - “Keep it deep. Keep it simple.”

GoogLeNet a.k.a. Inception Network

How to choose filter size?

- In convolutional neural networks, how to determine the optimal filter size?
- 3x3, 5x5, 7x7? (for example); which one to use in which layer?
- Idea: offer all, and let the network learn by itself

Inception Module

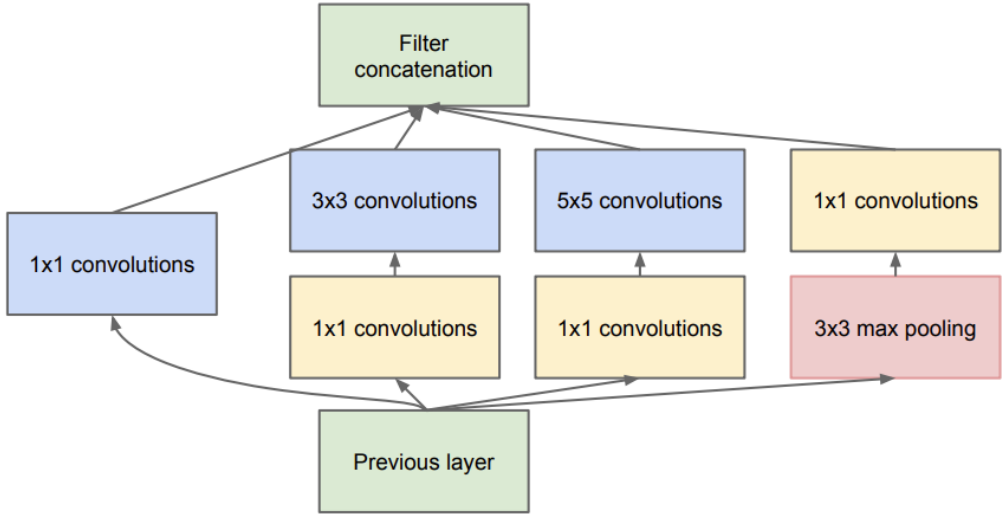


`https://arxiv.org/pdf/1409.4842.pdf`

(Szegedy et al., original paper)

- *Idea*: one layer looks like above
- During training, NN chooses way through filters by itself
- *Problem*: Non-negligible increase in parameters / channels

SOLUTION: 1X1 CONVOLUTION

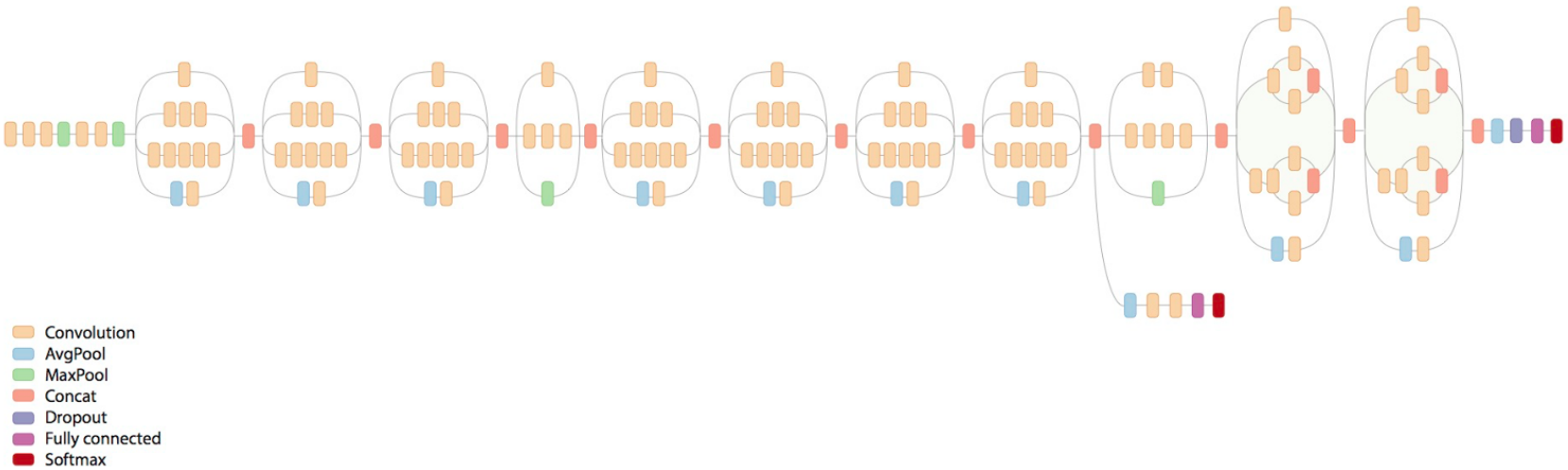


`https://arxiv.org/pdf/1409.4842.pdf`

(Szegedy et al., original paper)

- 1x1 convolution mends the issue
- For example, if 'previous layer' is of size $60 \times 100 \times 100$, filters are $60 \times 1 \times 1$; applying 20 such filters yields output of size $20 \times 100 \times 100$, a great reduction!
- See `https://www.youtube.com/watch?v=HunX473yXEI` for explanations

The Inception Network



Source: https://hackathonprojects.files.wordpress.com/2016/09/74911-image03.png

GoogLeNet

Features

- *Advantages*:
  - Top-5 accuracy: 93.4% (vs. 92.7% from VGG)
  - Performs various computational operations in parallel, and
  - stacks amazing 22 layers, while
  - being computationally reasonable: training needs a week
- *Why important*:
  - Points out that CNNs do not need to be stacked sequentially
  - Set the stage for further amazing architectures

GoogLeNet

Features

- *Advantages*:
  - Top-5 accuracy: 93.4% (vs. 92.7% from VGG)
  - Performs various computational operations in parallel, and
  - stacks amazing 22 layers, while
  - being computationally reasonable: training needs a week
- *Why important*:
  - Points out that CNNs do not need to be stacked sequentially
  - Set the stage for further amazing architectures

GoogLeNet

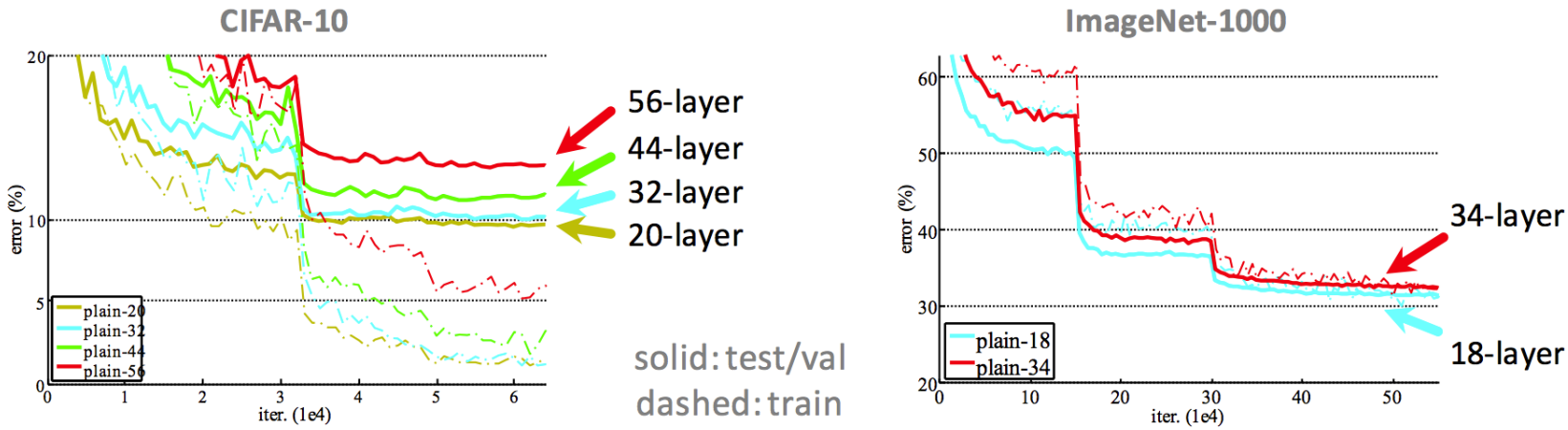
Features

- *Advantages*:
  - Top-5 accuracy: 93.4% (vs. 92.7% from VGG)
  - Performs various computational operations in parallel, and
  - stacks amazing 22 layers, while
  - being computationally reasonable: training needs a week
- *Why important*:
  - Points out that CNNs do not need to be stacked sequentially
  - Set the stage for further amazing architectures

ResNet

Motivation I

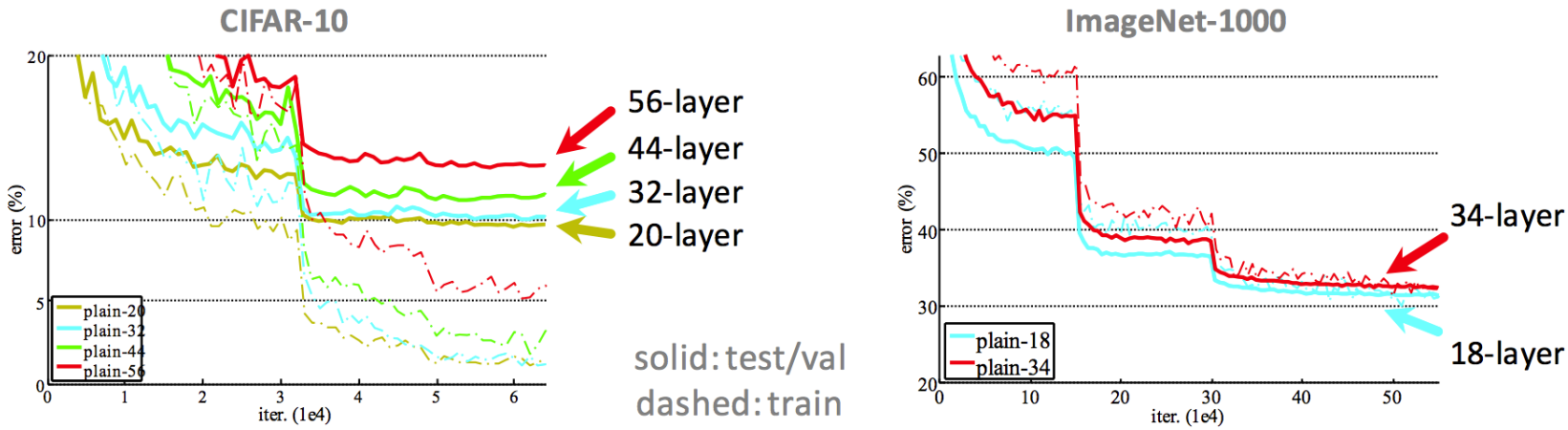
Are we ready to simply stack layers?



- Training(!) accuracy decreases on stacking layers
- Despite having solved the vanishing gradient problem
- What else could be the reason?

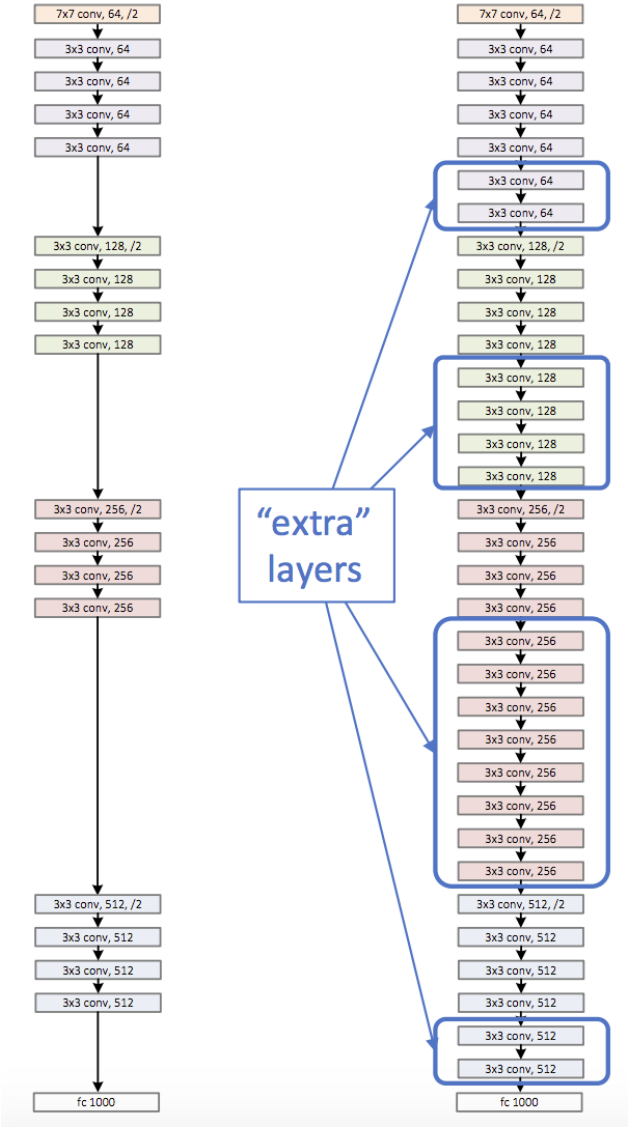
Motivation I

Are we ready to simply stack layers?



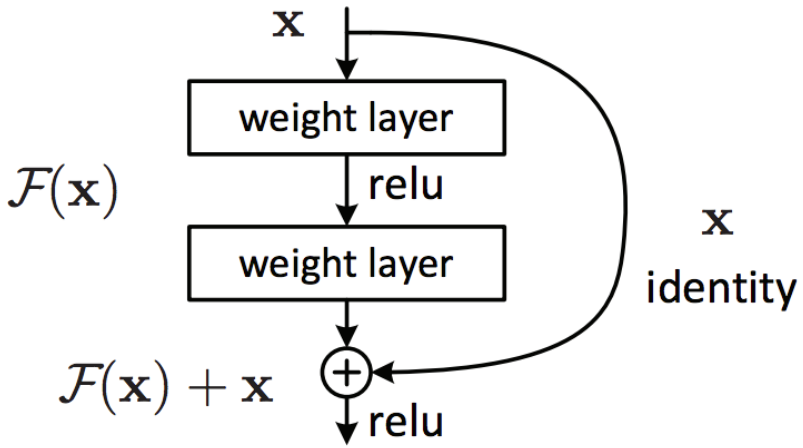
- Training(!) accuracy decreases on stacking layers
- Despite having solved the vanishing gradient problem
- What else could be the reason?

Motivation II



- Solution by construction:
  - Original layers: copied from shallower model
  - Extra layers: identity
  - Yet: training error increases in deeper model
- Solvers have trouble learning identity function
- In other words: if less layers are needed, deep model fails

ResNet Unit

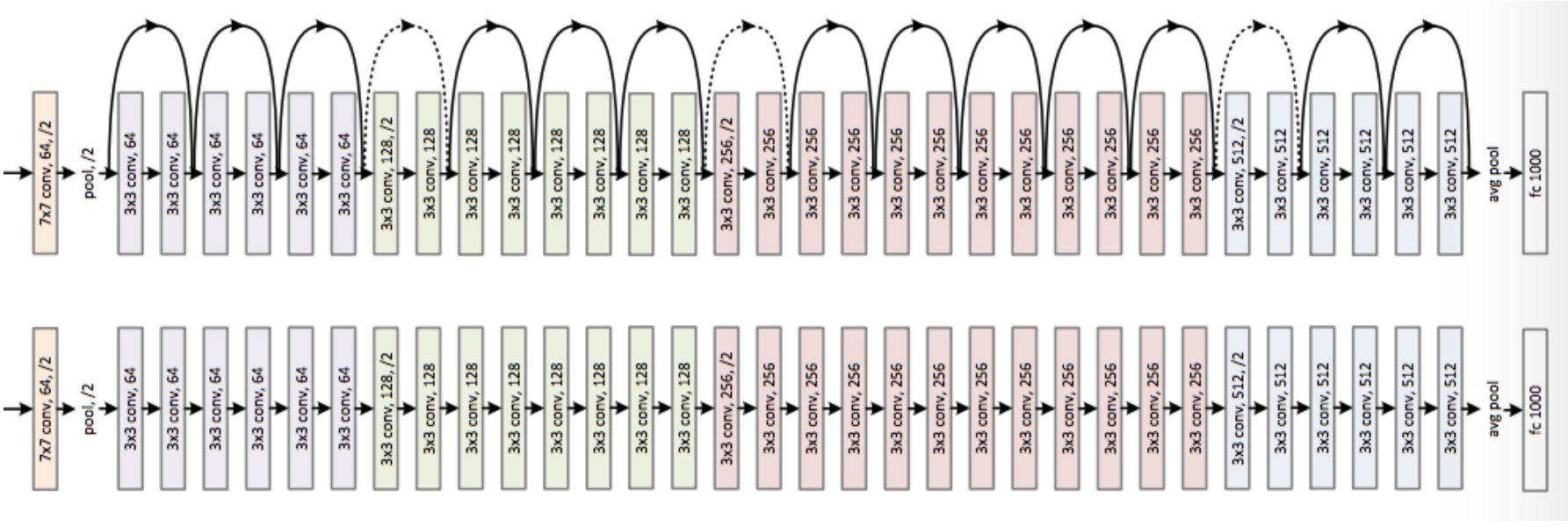


https://arxiv.org/pdf/1512.03385.pdf

(original paper)

- *Idea*: Bypass two layers by identity
- If identify function is to be learnt (or small modifications thereof)
  - $F(x)$ is to be learnt as zero
  - ☞ Easy for NN's to learn zero
- So: more layers can no longer harm if not needed!

Full Residual Network (ResNet)

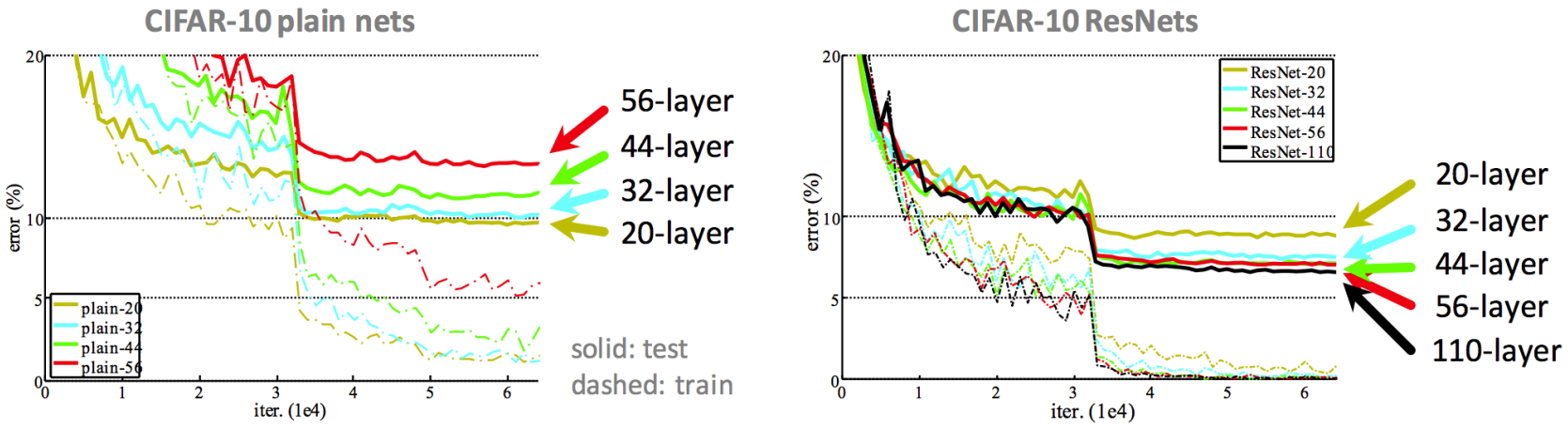


Plain network (bottom) vs. ResNet (top)

`https://arxiv.org/pdf/1512.03385.pdf`

(original paper)

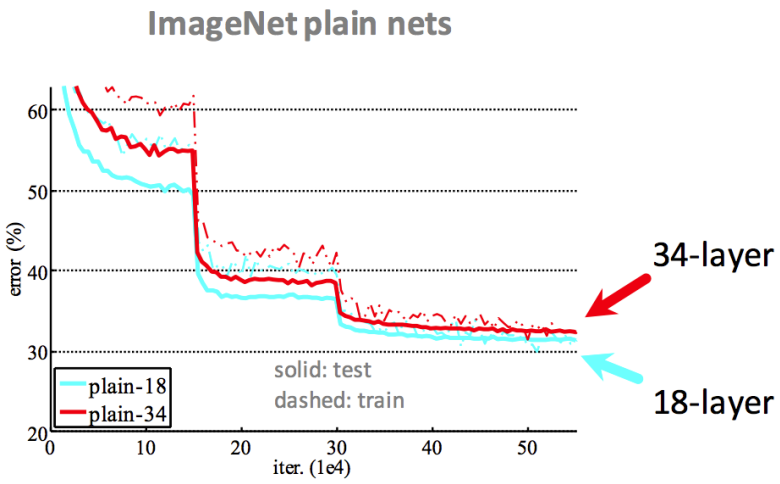
ResNet Results

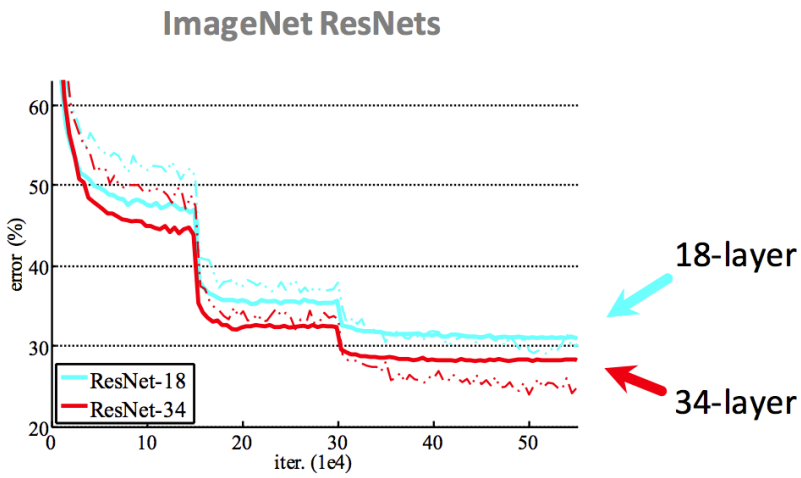


https://arxiv.org/pdf/1512.03385.pdf

(original paper)

ResNet Results





https://arxiv.org/pdf/1512.03385.pdf

(original paper)

ResNet

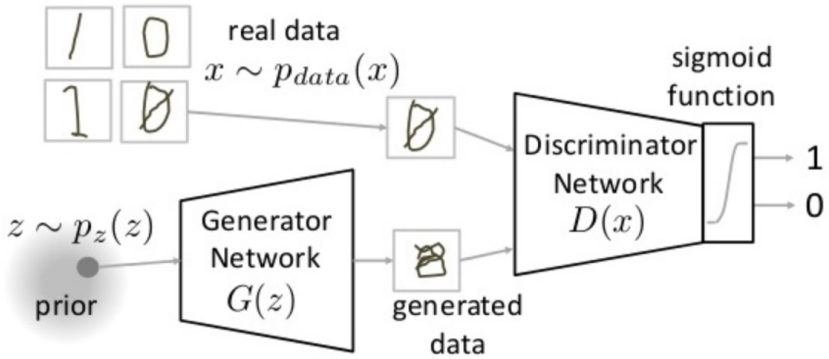
Features

- *Advantages*:
  - Top-5 accuracy: 96.4(!)% (vs. 93.4% from GoogLeNet)
  - "Ultra-deep": 152(!) layers
  - Just quite simply the best (as of 2016)
- *Why important*:
  - Breakthrough in going ultra-deep
  - Stacking ResNet units do not require extra performance upgrades (while necessary with ordinary CNNs)

Latest Advances

Latest Advances

Generative Adversarial Networks



The GAN generator aims to generate pictures

that the discriminator cannot distinguish from true pictures

- *Generative Adversarial Networks (GANs)*: Generative models, that is, they do not classify, but, for example, learn to draw pictures themselves
- *Resources*:
  - `https://skymind.ai/wiki/generative-adversarial-network-gan`
  - *Tutorial (by the inventor)*: `https://arxiv.org/pdf/1701.00160.pdf`
  - *Video (by the inventor)*: `https://www.youtube.com/watch?v=HGYYEUSm-0Q`

Latest Advances

Capsule Networks

- *Capsule Networks*:
  - Ordinary CNNs do not take translational and rotational relationships of elements of images into account
  - This can lead to terrible misclassification
  - Way out so far: pooling
  - Hinton:

    "The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster."

  - CapsNets do this better, by employing "capsules", not convolutional layers as central network element
- *Resources*:
  - *Brief Introduction*: `https://medium.com/ai3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b`
  - *Paper*: `https://arxiv.org/abs/1710.09829`
  - *Video Tutorial*: `https://www.youtube.com/watch?v=pPN8d0E3900`

Using the structure of genome data in the design of deep neural networks for predicting amyotrophic lateral sclerosis from genotype

Bojian Yin, Marleen Balvert, Rick van de Spek, Bas Dutilh, Sander Bohte, Jan Veldink, Alexander Schönhuth

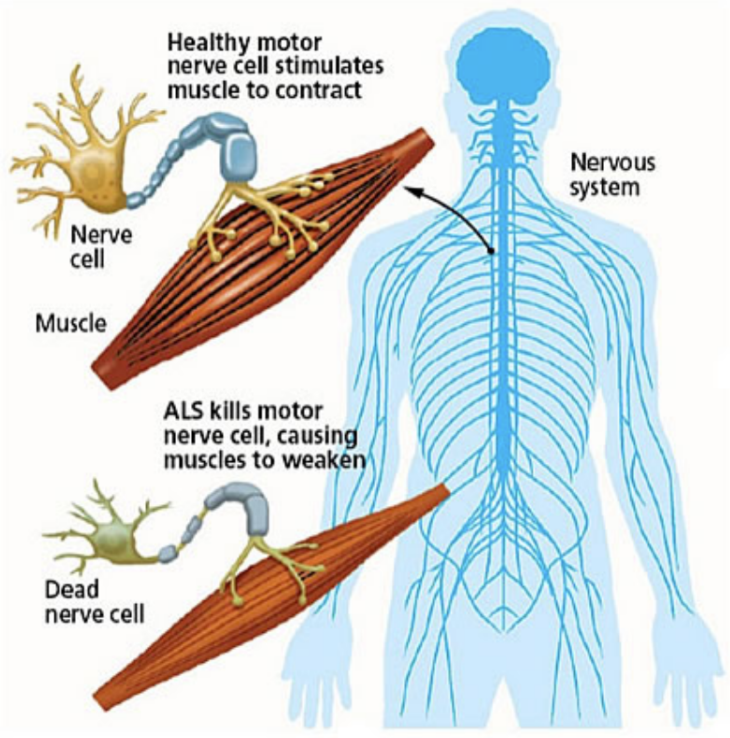
ISMB, Basel
July 24, 2019

Introduction

—

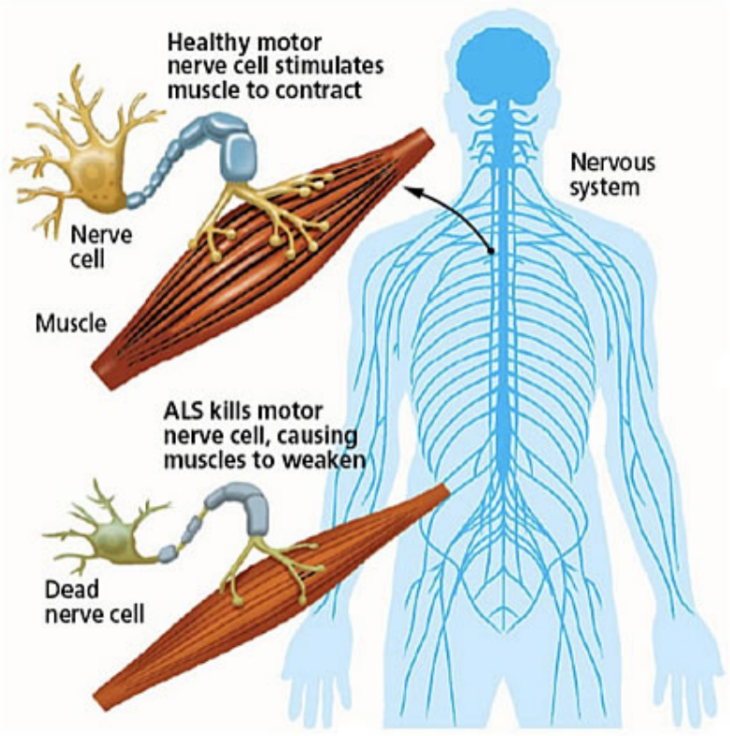
Amyotrophic Lateral Sclerosis

Amyotrophic Lateral Sclerosis (ALS)



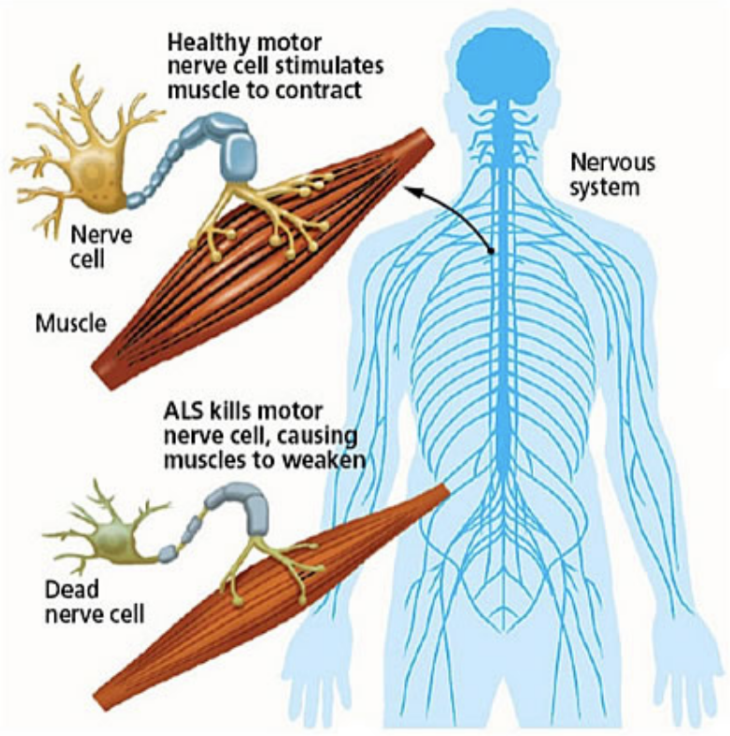
- "Motor Neuron Disease"
- Degenerates upper and lower motor neurons ☞ muscles weaken and shrink
- Prominent Patient: Stephen Hawking
- Death occurs two to five years after first symptoms (respiratory failure)
- Cumulative lifetime risk 1 in 300
- Point prevalence: 5 per 100 000

Amyotrophic Lateral Sclerosis (ALS)



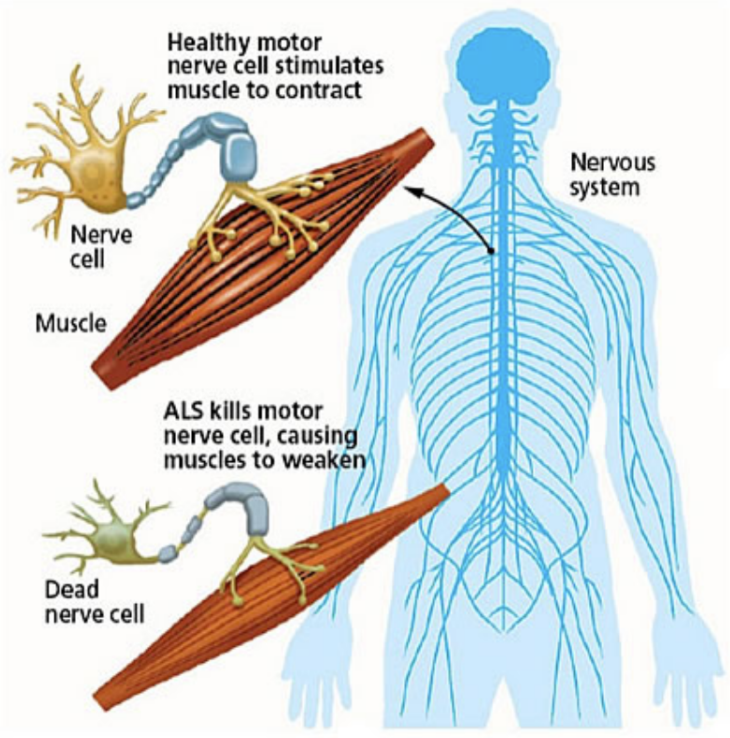
- "Motor Neuron Disease"
- Degenerates upper and lower motor neurons ☞ muscles weaken and shrink
- Prominent Patient: Stephen Hawking
- Death occurs two to five years after first symptoms (respiratory failure)
- Cumulative lifetime risk 1 in 300
- Point prevalence: 5 per 100 000

Amyotrophic Lateral Sclerosis (ALS)



- "Motor Neuron Disease"
- Degenerates upper and lower motor neurons ☞ muscles weaken and shrink
- Prominent Patient:
  Stephen Hawking
- Death occurs two to five years after first symptoms (respiratory failure)
- Cumulative lifetime risk 1 in 300
- Point prevalence: 5 per 100 000

Amyotrophic Lateral Sclerosis (ALS)



- "Motor Neuron Disease"
- Degenerates upper and lower motor neurons ☞ muscles weaken and shrink
- Prominent Patient: Stephen Hawking
- Death occurs two to five years after first symptoms (respiratory failure)
- Cumulative lifetime risk 1 in 300
- Point prevalence: 5 per 100 000

Very Poor Prognosis

ALS: Why Poor Prognosis?

- [van Rheenen et al., Nat Gen, 2016]
  - Twin studies: Heritability approximately 65%
  - Additive (SNP based) heritability: 8.5%

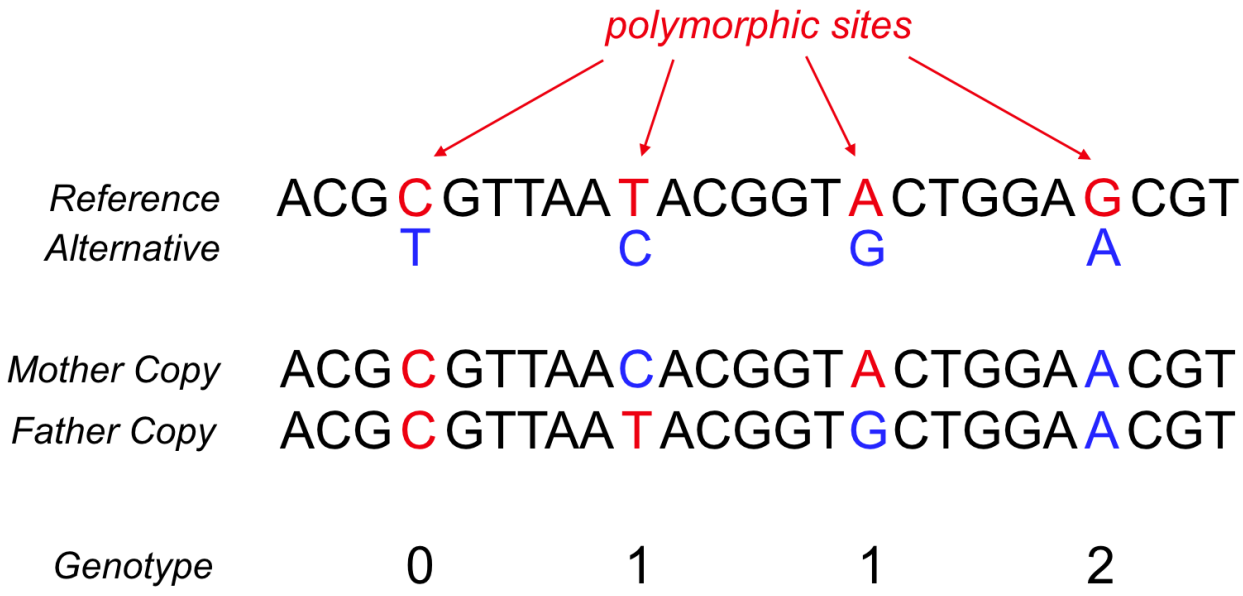
85% of heritability is still missing

- Explanation:
  - Non-additive, so far insufficiently understood relationships among genetic factors (epistasis), including rare variation establish ALS-related phenotype

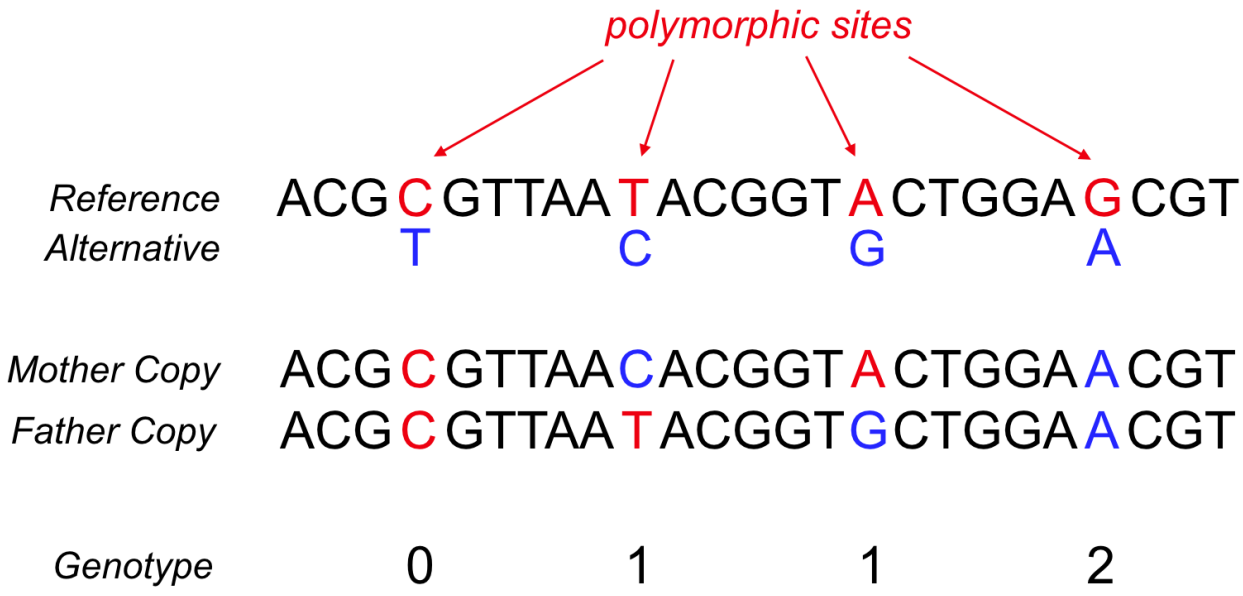
ALS has a complex genetic architecture

Genetic Architecture: A Formal Description

Individual Genotypes



Individual Genotypes



Represent individuals by their genotypes

- Vectors whose length is number of polymorphic sites, with entries
  - 0 = homozygous for reference
  - 1 = heterozygous
  - 2 = homozygous for alternative

The Genetic Architecture of ALS

Definition

Let X be all people, represented by their genotypes.

The *genetic architecture* f_{ALS} of ALS is a function

$$f_{\text{ALS}} : X \longrightarrow \{0, 1\}$$

The Genetic Architecture of ALS

Definition

Let X be all people, represented by their genotypes.

The *genetic architecture* f_{ALS} of ALS is a function

$$f_{\text{ALS}} : X \longrightarrow \{0, 1\}$$

where

$$f(x) = \begin{cases} 1 & x \text{ affected by ALS} \\ 0 & \text{otherwise} \end{cases}$$

Machine Learning the Genetic Architecture

Learning the Genetic Architecture

Let $\mathcal{M}$ is a class of (ML compatible) functions, such as SVM's or Deep Neural Networks:

$$\text{Approximate } f_{\text{ALS}} \text{ by } f^*_{\text{ALS}} \in \mathcal{M}$$

- using known examples
  - cases: $(x, f_{\text{ALS}}(x) = 1)$
  - controls: $(x, f_{\text{ALS}}(x) = 0)$

  as *training/validation data*
- Goodness of $f^*_{\text{ALS}}(x)$ is evaluated on previously unseen *test data*

Learning the Genetic Architecture

Let $\mathcal{M}$ is a class of (ML compatible) functions, such as SVM's or Deep Neural Networks:

$$\text{Approximate } f_{\text{ALS}} \text{ by } f^*_{\text{ALS}} \in \mathcal{M}$$

- using known examples
  - cases: $(x, f_{\text{ALS}}(x) = 1)$
  - controls: $(x, f_{\text{ALS}}(x) = 0)$

  as *training/validation data*
- Goodness of $f^*_{\text{ALS}}(x)$ is evaluated on previously unseen *test data*

Learning the Genetic Architecture

Let $\mathcal{M}$ is a class of (ML compatible) functions, such as SVM's or Deep Neural Networks:

$$\text{Approximate } f_{\text{ALS}} \text{ by } f^*_{\text{ALS}} \in \mathcal{M}$$

- using known examples
  - cases: $(x, f_{\text{ALS}}(x) = 1)$
  - controls: $(x, f_{\text{ALS}}(x) = 0)$

  as *training/validation data*
- Goodness of $f^*_{\text{ALS}}(x)$ is evaluated on previously unseen *test data*

First Question
Has this been tried before?

Machine Learning the Genetic Architecture

Tried Before? – Yes and No

Yes

- GWAS try to approximate the genetic architecture f_{ALS} by additive (linear) functions $f^*_{\text{ALS}} : X \longrightarrow \{0, 1\}$
- *Advantage*: f^*_{ALS} easy to perceive
- *However*, missing heritability:
  - Linear/additive approximation is poor
  - Complex, non-additive approximations needed

Machine Learning the Genetic Architecture

Tried Before? – Yes and No

Yes

- GWAS try to approximate the genetic architecture f_{ALS} by additive (linear) functions $f^*_{\text{ALS}} : X \longrightarrow \{0, 1\}$
- *Advantage*: f^*_{ALS} easy to perceive
- *However*, missing heritability:
  - Linear/additive approximation is poor
  - Complex, non-additive approximations needed

No

- Manual (GWAS type) conception of non-additive approximations has proven to be notoriously challenging
- Genuine machine learning settings have never been tried

Machine Learning the Genetic Architecture

Tried Before? – Yes and No

Yes

- GWAS try to approximate the genetic architecture f_{ALS} by additive (linear) functions $f^*_{\text{ALS}} : X \longrightarrow \{0, 1\}$
- *Advantage*: f^*_{ALS} easy to perceive
- *However*, missing heritability:
  - Linear/additive approximation is poor
  - Complex, non-additive approximations needed

No

- Manual (GWAS type) conception of non-additive approximations has proven to be notoriously challenging
- Genuine machine learning settings have never been tried

Second Question
Is there sufficient training data available?
Yes: Project MinE.

ALS Genotype Data: Project MinE

- See `www.projectmine.com` (donate?)
- Target: sequencing 22 500 individuals, of which 15 000 cases
- Genotype data available from
  - up to 23 000 ALS patients
  - more than 80 000 controls
- At our disposal: Dutch cohort, 11525 individuals, of which
  - 4411 cases $\leftrightarrow$ examples $(x, f_{\text{ALS}}(x) = 1)$
  - 7114 controls $\leftrightarrow$ examples $(x, f_{\text{ALS}}(x) = 0)$

Sufficient (training) data for high-performance Machine Learning

Machine Learning the Genetic Architecture

Of the classes of functions $\mathcal{M}$ we tried, Deep Neural Networks

- came with the most theoretical promises
- were most challenging to implement
- outperformed all other approaches

Machine Learning the Genetic Architecture

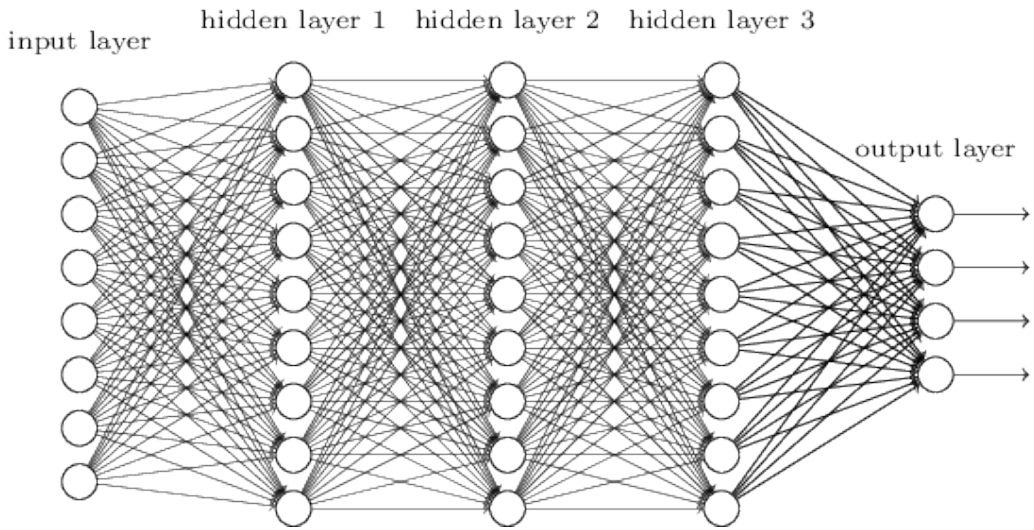
Of the classes of functions $\mathcal{M}$ we tried, Deep Neural Networks

- came with the most theoretical promises
- were most challenging to implement
- outperformed all other approaches

Deep Learning – Promises and Challenges

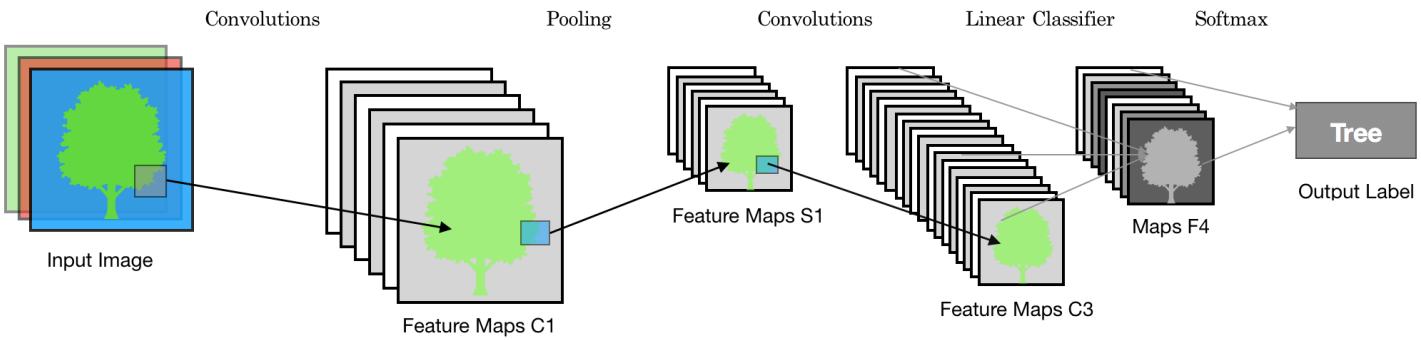
Feedforward Neural Networks

Deep Learning



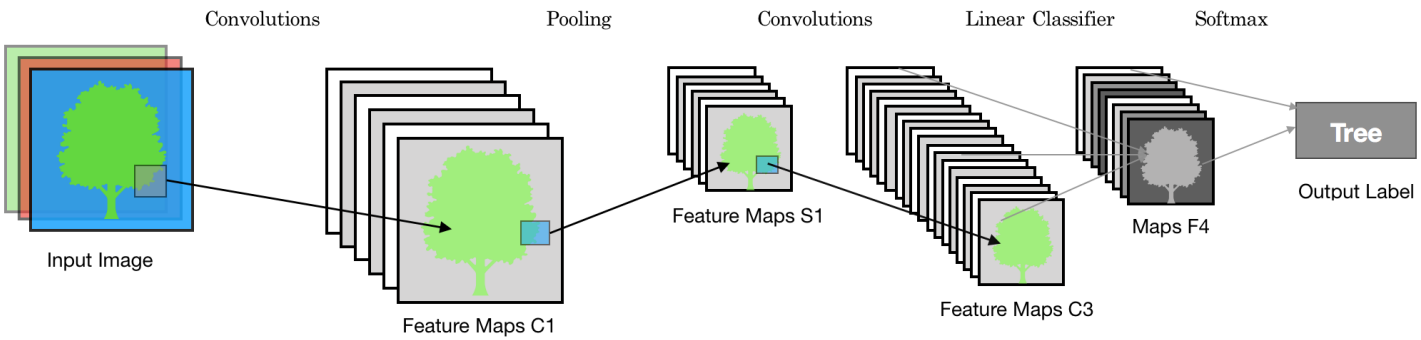
- *Neuron (Circle)*: Linear operation + activation function
- *Depth of Network*: Number of hidden layers; "deep" $\leftrightarrow$ at least 8 layers (historical reasons)
- *Theoretical Promise* [*Universal Approximation Theorem*; Hornik et al., 1989; Cybenko, 1989; Montufar, 2014]: *Arbitrarily accurate approximation where accuracy increases exponentially in depth*: the deeper the better.

Convolutional Neural Networks (CNNs)



- Convolution filters identify substructures and, layer by layer, combine them to reach final conclusions
- Origin of Deep Learning ['AlexNet', 2012]: *Superhuman performance* in image classification, CNN with 8 hidden layers

Convolutional Neural Networks (CNNs)



- Convolution filters identify substructures and, layer by layer, combine them to reach final conclusions
- Origin of Deep Learning ['AlexNet', 2012]: *Superhuman performance* in image classification, CNN with 8 hidden layers

Promise / Motivation

—

Superhuman performance in "understanding" $\mathbf{f}_{\mathrm{ALS}}$*?*

Challenge

—

Make CNN of depth at least 8 work for genotypes

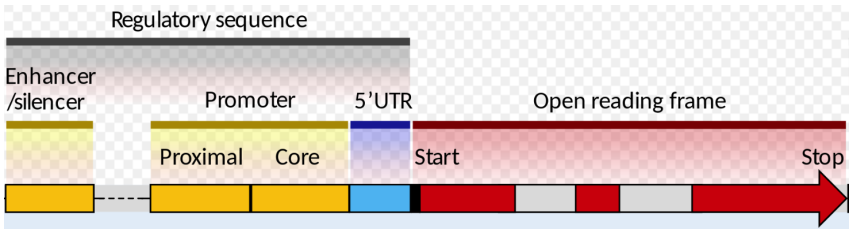
Feature Selection

—

Picking relevant variant sites from 5 million sites overall

Feature Selection: Two-Step Procedure

- *First Step:* Select 64 sites (56 up-, 8 downstream of TSS) from each promoter region; discard all others



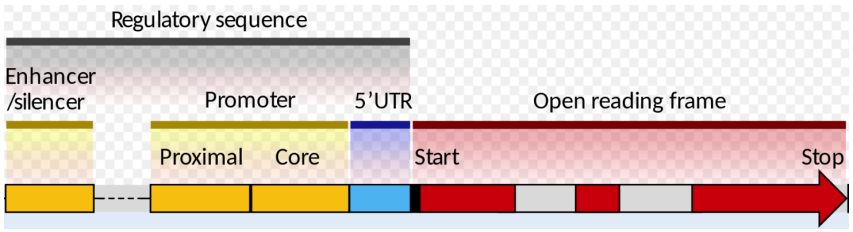
Regulatory Region Gene

  - Majority of disease-associated variants sit in regulatory regions [Maurano et al., Science, 2012]
  - ***Remaining sites*: more than a million**

- *Second Step*: Evaluate promoter variant blocks using (5-layered) 'Promoter-CNN' for being predictive of ALS
  - keep only 32 highest-scoring regions; discard all others
  - ***Remaining sites*: 2048 ☞ Perfect!**

Feature Selection: Two-Step Procedure

- *First Step:* Select 64 sites (56 up-, 8 downstream of TSS) from each promoter region; discard all others



Regulatory Region Gene

  - Majority of disease-associated variants sit in regulatory regions [Maurano et al., Science, 2012]
  - ***Remaining sites*: more than a million**

- *Second Step*: Evaluate promoter variant blocks using (5-layered) 'Promoter-CNN' for being predictive of ALS

  - keep only 32 highest-scoring regions; discard all others
  - ***Remaining sites*: 2048 ☞ Perfect!**

Feature Selection: Additional Advantages

- Applying convolution sensible: By laws of recombination, variants within promoter region inherited as haplotype block
- Interpretability: Reveal potentially relevant ALS genes

Feature Selection: Additional Advantages

- Applying convolution sensible: By laws of recombination, variants within promoter region inherited as haplotype block
- Interpretability: Reveal potentially relevant ALS genes

Feature Selection: Additional Advantages

- Applying convolution sensible: By laws of recombination, variants within promoter region inherited as haplotype block
- Interpretability: Reveal potentially relevant ALS genes

Feature selection addresses three key technical challenges

Feature Selection: Additional Advantages

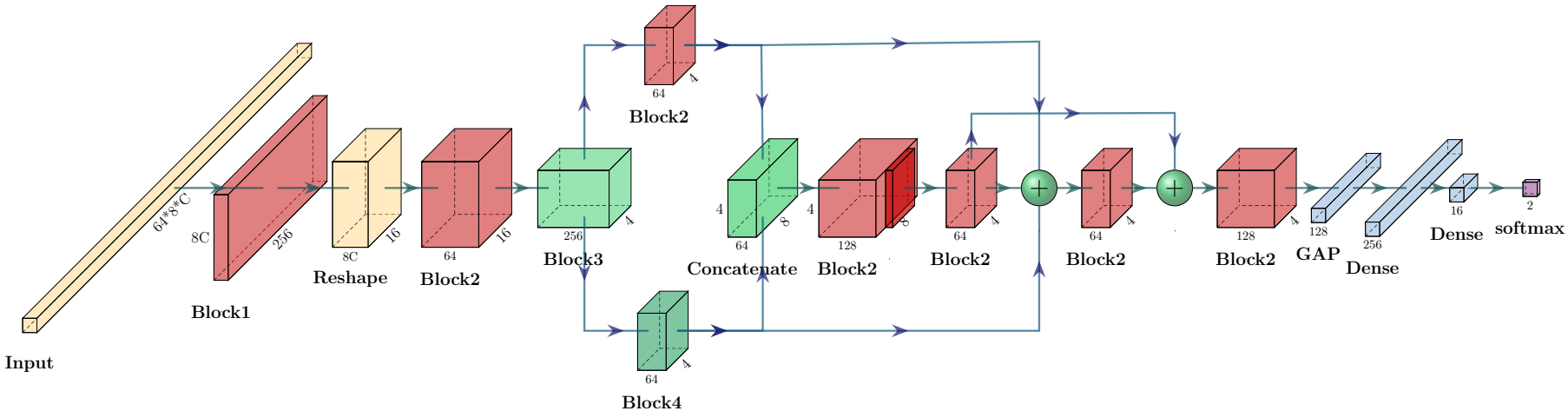
- Applying convolution sensible: By laws of recombination, variants within promoter region inherited as haplotype block
- Interpretability: Reveal potentially relevant ALS genes

Left To Do

Construct CNN with at least 8 hidden layers

Classification: ALS-Net

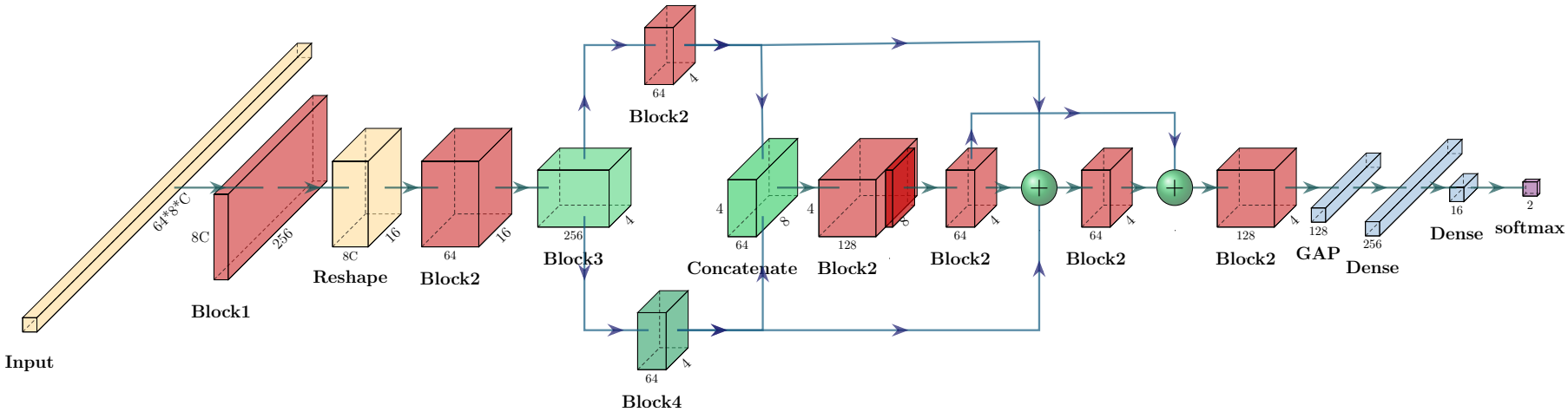
ALS-Net: Architecture



Depth: 24 hidden layers overall

- *Block 1*: 2 x conv. + batch norm. (BN), *Block 2*: 3 x conv.
- *Reshape*: [Howard et al., 2017]: yields $16 \times 16 \times 32$ ☞ convolution
- *Block 3*: 1 x sep. conv. [Gao et al., 2018]: saves on parameters, 1 x conv., 2 x pooling
- *Block 4*: 2 x conv. + 1 x separable conv.
- *Blue Arrows*: Bypass layers if needed, see [ResNet, 2015]
- *GAP*: Global average pooling
- *Dense*: Fully connected layer

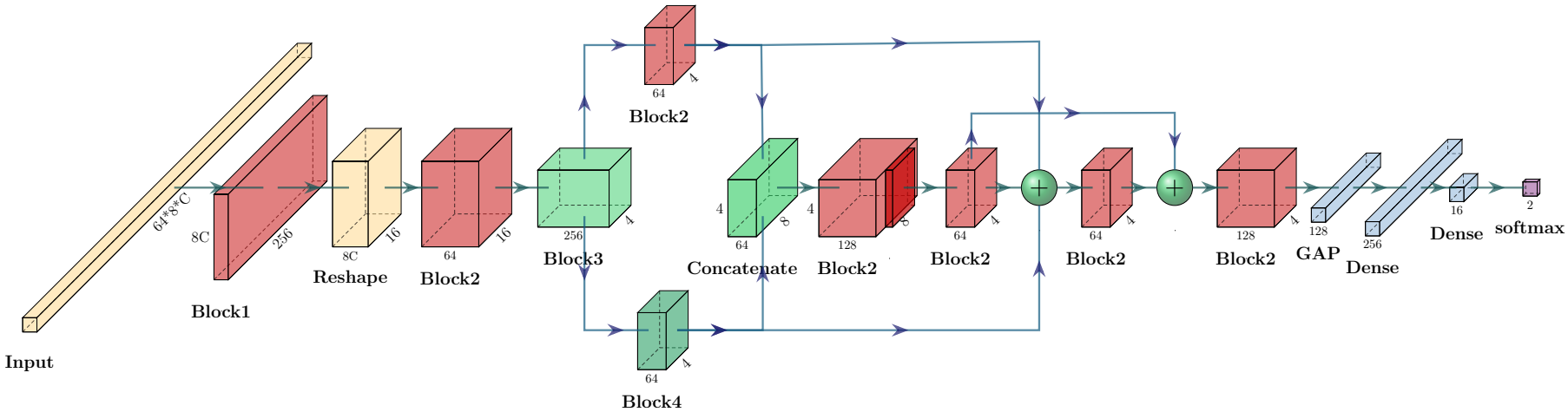
ALS-Net: Architecture



Depth: 24 hidden layers overall

- *Block 1*: 2 x conv. + batch norm. (BN), *Block 2*: 3 x conv.
- *Reshape*: [Howard et al., 2017]: yields $16 \times 16 \times 32$ ☞ convolution
- *Block 3*: 1 x sep. conv. [Gao et al., 2018]: saves on parameters, 1 x conv., 2 x pooling
- *Block 4*: 2 x conv. + 1 x separable conv.
- *Blue Arrows*: Bypass layers if needed, see [ResNet, 2015]
- *GAP*: Global average pooling
- *Dense*: Fully connected layer

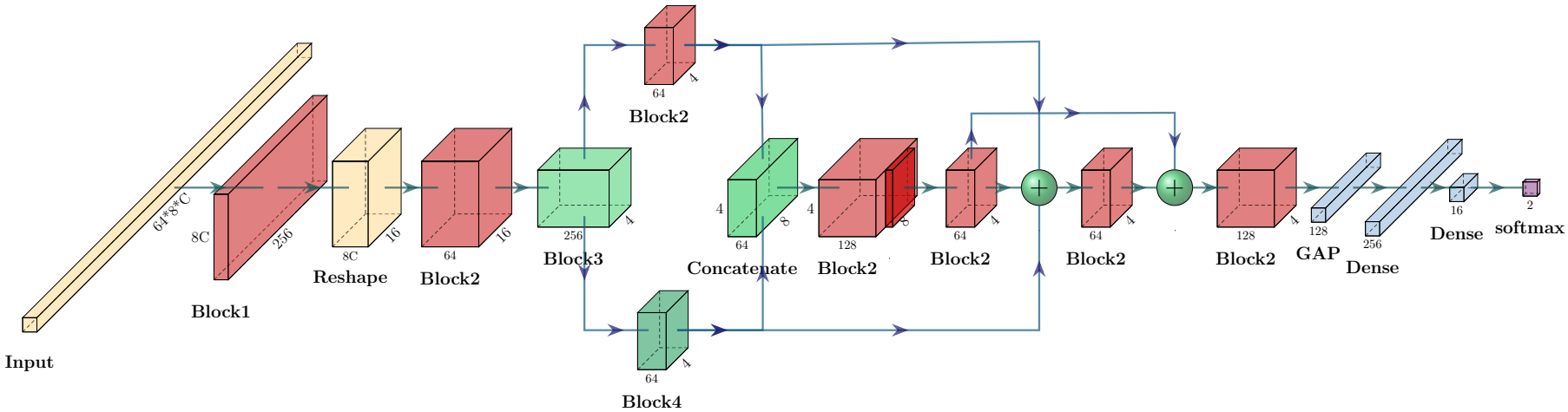
ALS-Net: Architecture



Depth: 24 hidden layers overall

- *Block 1*: 2 x conv. + batch norm. (BN), *Block 2*: 3 x conv.
- *Reshape*: [Howard et al., 2017]: yields $16 \times 16 \times 32$ ☞ convolution
- *Block 3*: 1 x sep. conv. [Gao et al., 2018]: saves on parameters, 1 x conv., 2 x pooling
- *Block 4*: 2 x conv. + 1 x separable conv.
- *Blue Arrows*: Bypass layers if needed, see [ResNet, 2015]
- *GAP*: Global average pooling
- *Dense*: Fully connected layer

ALS-Net: Architecture



Depth: 24 hidden layers overall

- *Block 1*: 2 x conv. + batch norm. (BN), *Block 2*: 3 x conv.
- *Reshape*: [Howard et al., 2017]: yields $16 \times 16 \times 32$ ☞ convolution
- *Block 3*: 1 x sep. conv. [Gao et al., 2018]: saves on parameters, 1 x conv., 2 x pooling
- *Block 4*: 2 x conv. + 1 x separable conv.
- *Blue Arrows*: Bypass layers if needed, see [ResNet, 2015]
- *GAP*: Global average pooling
- *Dense*: Fully connected layer

Chromosomes 7, 9, 17 and 22

	Accuracy	Precision	Recall
Logistic Regression	73.9	75.9	69.9
Support Vector Machines	72.5	78.3	62.4
Random Forest	59.6	**81.3**	24.9
Ada-Boost	66.1	70.0	56.5
ALS-Net	**76.9**	71.1	**90.8**

- Recall: ALS-Net recovers substantially more cases
- Choice of chromosomes favors additive approaches [Van Rheenen et al., Nat.Gen., 2016]
- All methods required (here: CNN-based) feature selection:
- *Important*: ALS-Net picks up less confounding variables (batch effects) than Logistic Regression (see 4.5 in the paper)

Chromosomes 7, 9, 17 and 22

	Accuracy	Precision	Recall
Logistic Regression	73.9	75.9	69.9
Support Vector Machines	72.5	78.3	62.4
Random Forest	59.6	**81.3**	24.9
Ada-Boost	66.1	70.0	56.5
ALS-Net	**76.9**	71.1	**90.8**

- Recall: ALS-Net recovers substantially more cases
- Choice of chromosomes favors additive approaches [Van Rheenen et al., Nat.Gen., 2016]
- All methods required (here: CNN-based) feature selection:
- *Important*: ALS-Net picks up less confounding variables (batch effects) than Logistic Regression (see 4.5 in the paper)

Chromosomes 7, 9, 17 and 22

	Accuracy	Precision	Recall
Logistic Regression	73.9	75.9	69.9
Support Vector Machines	72.5	78.3	62.4
Random Forest	59.6	**81.3**	24.9
Ada-Boost	66.1	70.0	56.5
ALS-Net	**76.9**	71.1	**90.8**

- Recall: ALS-Net recovers substantially more cases
- Choice of chromosomes favors additive approaches [Van Rheenen et al., Nat.Gen., 2016]
- All methods required (here: CNN-based) feature selection:
- *Important*: ALS-Net picks up less confounding variables (batch effects) than Logistic Regression (see 4.5 in the paper)

Chromosomes 7, 9, 17 and 22

	Accuracy	Precision	Recall
Logistic Regression	73.9	75.9	69.9
Support Vector Machines	72.5	78.3	62.4
Random Forest	59.6	**81.3**	24.9
Ada-Boost	66.1	70.0	56.5
ALS-Net	**76.9**	71.1	**90.8**

- Recall: ALS-Net recovers substantially more cases
- Choice of chromosomes favors additive approaches [Van Rheenen et al., Nat.Gen., 2016]
- All methods required (here: CNN-based) feature selection:
- *Important*: ALS-Net picks up less confounding variables (batch effects) than Logistic Regression (see 4.5 in the paper)

Outlook

- Saliency maps measure relevance of variant sites relative to influencing prediction:
  - Approaching TSS: only cases sensitive to changes
  - *Can we make sense of this?*
- Varying Data:
  - Other chromosomes
  - Haplotype not genotype data
  - Integrating RNA and epigenome data
  - Selecting different regions (e.g. with DeepSEA [Zhou, Troyanskaya, Nat.Met., 2015])
- Many of the selected genes so far not considered
- Approach is generic: what about other diseases?

Outlook

- Saliency maps measure relevance of variant sites relative to influencing prediction:
  - Approaching TSS: only cases sensitive to changes
  - *Can we make sense of this?*
- Varying Data:
  - Other chromosomes
  - Haplotype not genotype data
  - Integrating RNA and epigenome data
  - Selecting different regions (e.g. with DeepSEA [Zhou, Troyanskaya, Nat.Met., 2015])
- Many of the selected genes so far not considered
- Approach is generic: what about other diseases?

Outlook

- Saliency maps measure relevance of variant sites relative to influencing prediction:
  - Approaching TSS: only cases sensitive to changes
  - *Can we make sense of this?*
- Varying Data:
  - Other chromosomes
  - Haplotype not genotype data
  - Integrating RNA and epigenome data
  - Selecting different regions (e.g. with DeepSEA [Zhou, Troyanskaya, Nat.Met., 2015])
- Many of the selected genes so far not considered
- Approach is generic: what about other diseases?

Outlook

- Saliency maps measure relevance of variant sites relative to influencing prediction:
  - Approaching TSS: only cases sensitive to changes
  - *Can we make sense of this?*
- Varying Data:
  - Other chromosomes
  - Haplotype not genotype data
  - Integrating RNA and epigenome data
  - Selecting different regions (e.g. with DeepSEA [Zhou, Troyanskaya, Nat.Met., 2015])
- Many of the selected genes so far not considered
- Approach is generic: what about other diseases?

Summary

- *AI approaches for learning the genetic architecture of ALS*
- *Leveraged convolution to work for processing genotype data*
- *GWAS inspired protocol for enhanced interpretability*
- *Deep CNN's outperformed all other approaches*

Summary

- *AI approaches for learning the genetic architecture of ALS*
- *Leveraged convolution to work for processing genotype data*
- *GWAS inspired protocol for enhanced interpretability*
- *Deep CNN's outperformed all other approaches*

Summary

- *AI approaches for learning the genetic architecture of ALS*
- *Leveraged convolution to work for processing genotype data*
- *GWAS inspired protocol for enhanced interpretability*
- *Deep CNN's outperformed all other approaches*

Summary

- *AI approaches for learning the genetic architecture of ALS*
- *Leveraged convolution to work for processing genotype data*
- *GWAS inspired protocol for enhanced interpretability*
- *Deep CNN's outperformed all other approaches*

Summary

- *AI approaches for learning the genetic architecture of ALS*
- *Leveraged convolution to work for processing genotype data*
- *GWAS inspired protocol for enhanced interpretability*
- *Deep CNN's outperformed all other approaches*

First step into promising direction

Acknowledgments

- Stichting ALS
- Project MinE
- *First authors*: Bojian Yin, Marleen Balvert (CWI Amsterdam)
- Jan Veldink, Rick van de Spek (UMC Utrecht)
- Sander Bohte (Machine Learning Group, CWI Amsterdam)
- Bas Dutilh (U Utrecht)

Summary

- Recurrent Networks
  - http://www.deeplearningbook.org/, Chapter 10
  - http://neuralnetworksanddeeplearning.com/, Chapter 6, "Other approaches to deep neural nets"
- Going Deep
  - http://neuralnetworksanddeeplearning.com/, Chapter 6, "Recent progress in image recognition"
- Recent Advances
  - Generative Adversarial Networks: http://www.deeplearningbook.org/, 20.10.4
  - Capsule Networks: https://medium.com/aiC2B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-in

Thanks for your attention