CS109 – Data Science
Deep Learning III – Advanced Techniques

Hanspeter Pfister, Mark Glickman, Verena Kaynig-Fittkau

https://pt.slideshare.net/SergeyShelpuk/machine-learning-advanced-topics-overview

Announcements

- Alyssa now is Dr. Wilson!!

- Wednesday class is in Yenching auditorium
- And given by Sasha Rush!
- It is going to be about RNNs

Your Toolbox So Far

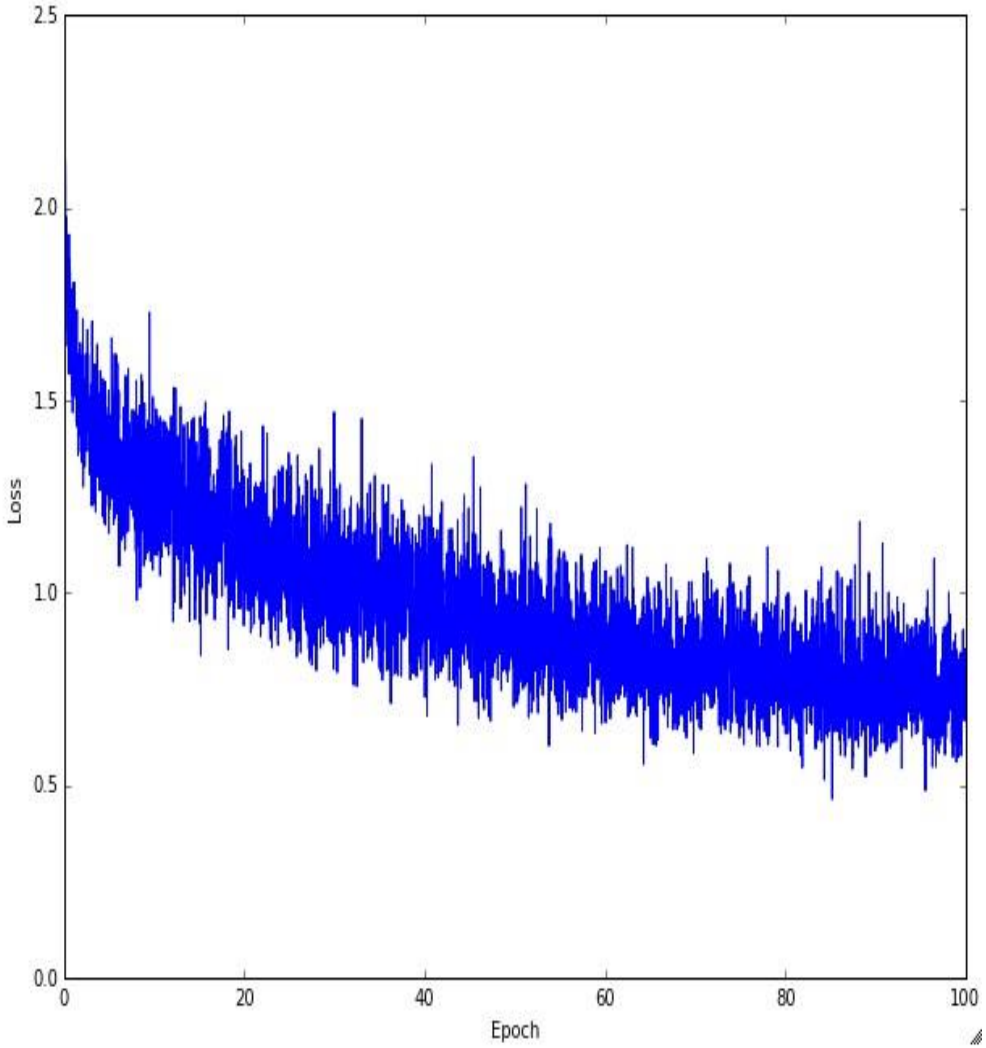
- What we have so far:
  - CNN layer
  - MaxPool layer
  - FC layer
  - BatchNorm Layer
  - Activations:
    - ReLU
    - sigmoid or softmax for last layer
  - Cost function – cross entropy
  - Initialization – Glorot or He
  - Training – Adam
  - Regularization – dropout + L2/L1
  - Convergence check – Early stopping
  - Data Preprocessing and Augmentation

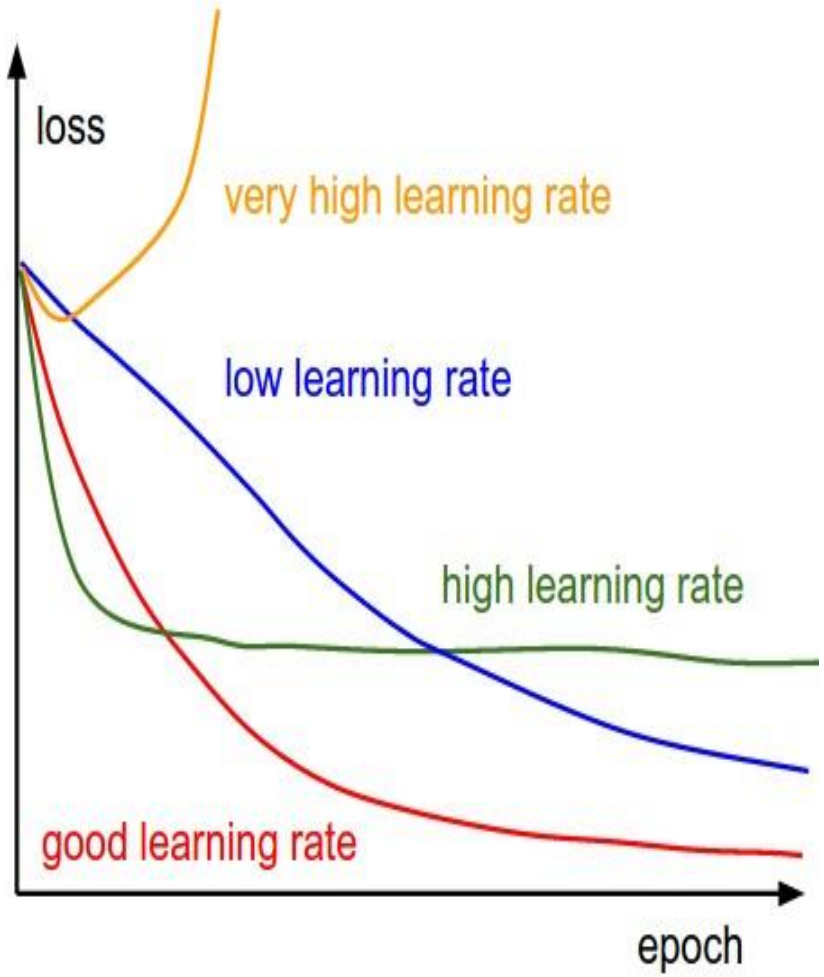
Learning Rate (once again)

- Need to find good range for learning rate
  - Look at the loss during training
  - If it is NaN => learning rate too high
  - If it decreases too slow => learning rate too low

- Also remember batch size and gradient estimation influence
  - Large batch – large learning rate – fewer updates
  - Small batch – smaller learning rate – more updates

Monitor and visualize the loss curve



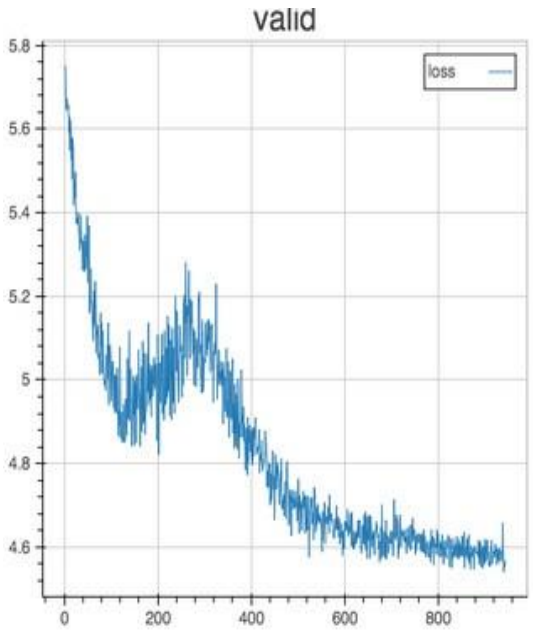


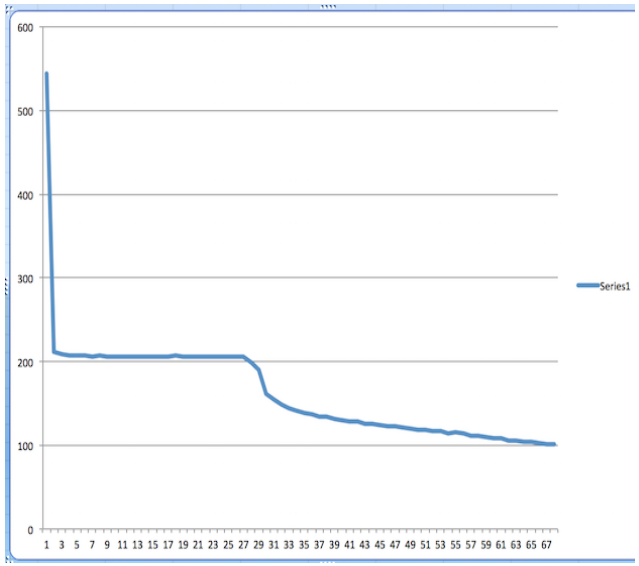
http://cs231n.github.io/

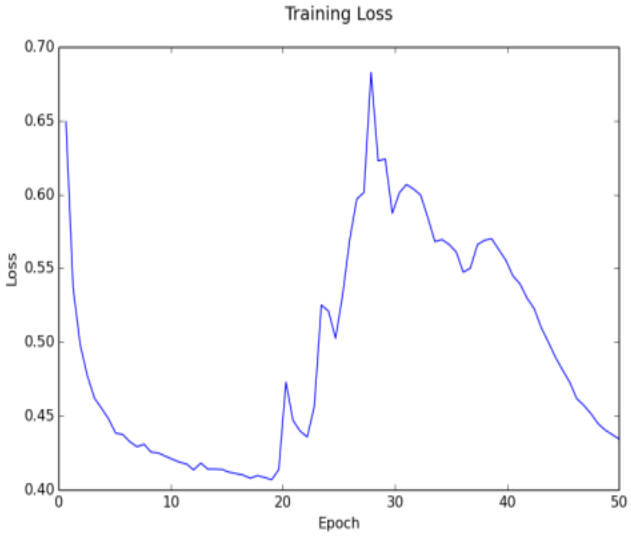
lossfunctions.tumblr.com

Loss function specimen

Andrey Karpathy





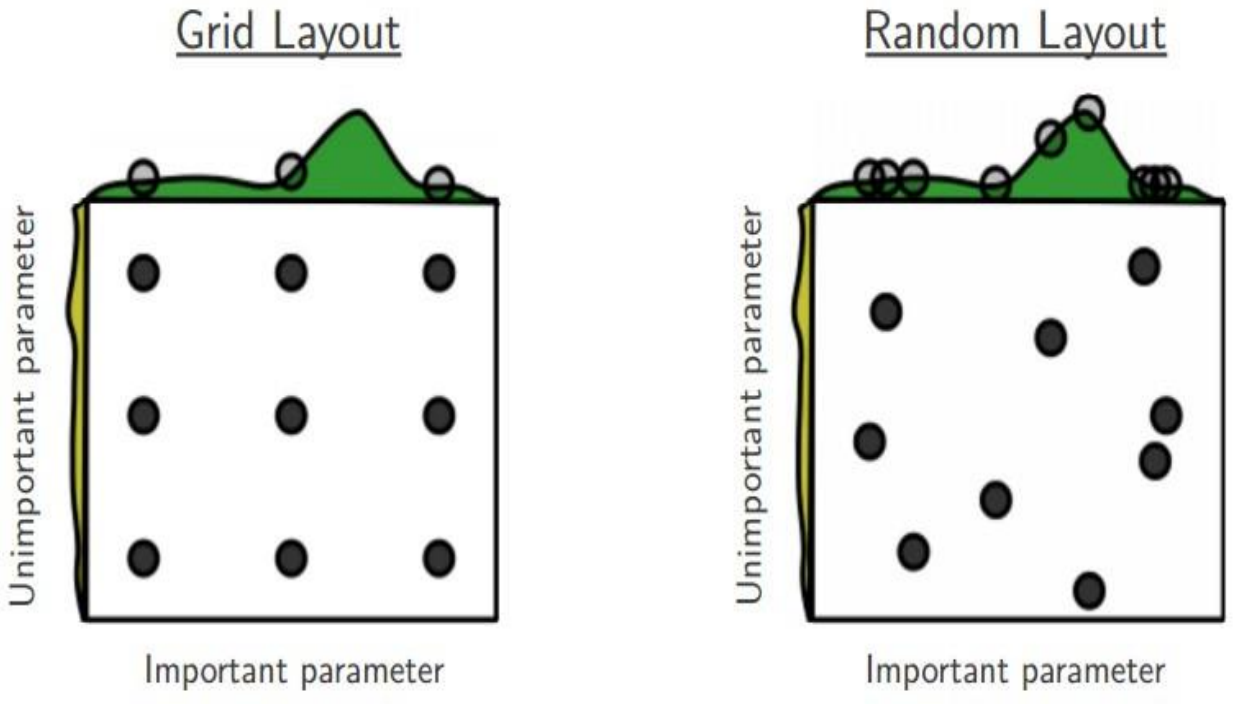


http://cs231n.github.io/

Cross Validation for Parameter Tuning

- How to fine tune the learning rate and regularization?
- Just like any classifier you can also use CV for deep learning
- Costs for trying a parameter setting are high
- People tend to use coarse grids
- This can be bad

Random Search vs. Grid Search



Random Search for Hyper-Parameter Optimization
Bergstra and Bengio, 2012

Architecture Design

- Things to consider:
  - Previous work done on similar problems
  - Computational resources

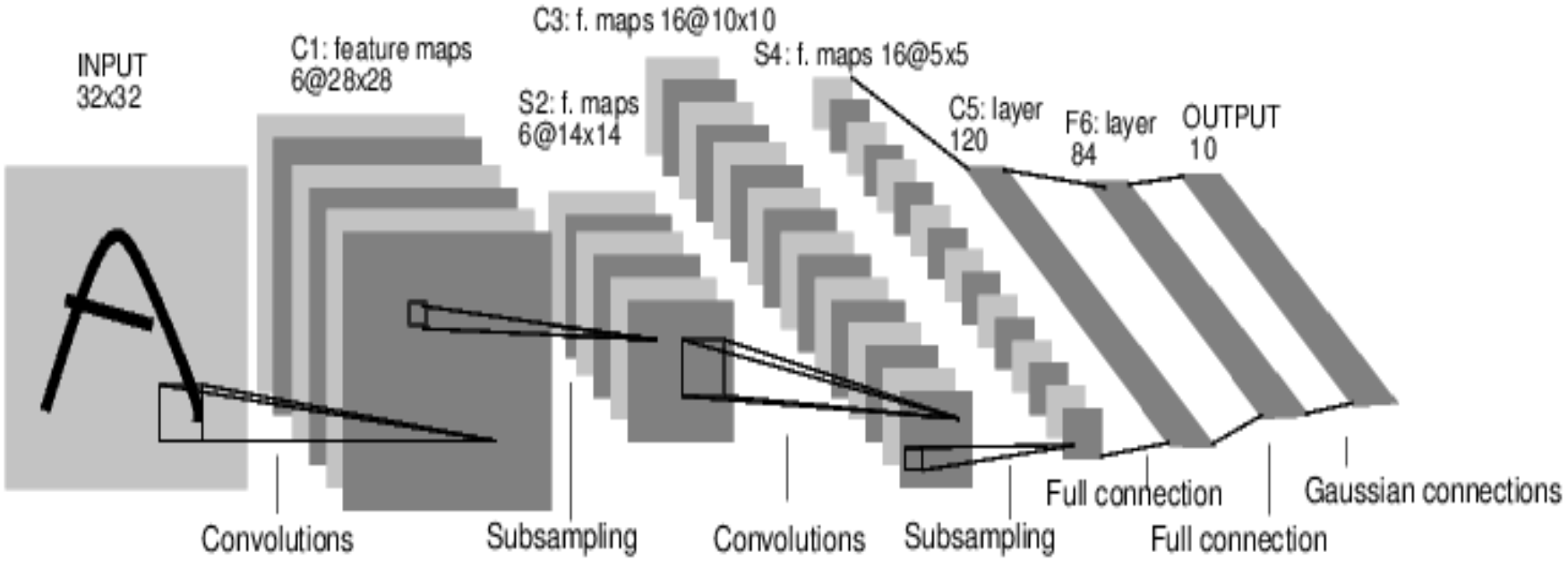
- Wider, and deeper means more degrees of freedom
  - Better classification performance
  - IF you can tune it right

Case studies

- Don‘t go from scratch when you learn deep learning
- Look at example models
- Use tricks that have been shown to work

Case Study: LeNet-5

[LeCun et al., 1998]

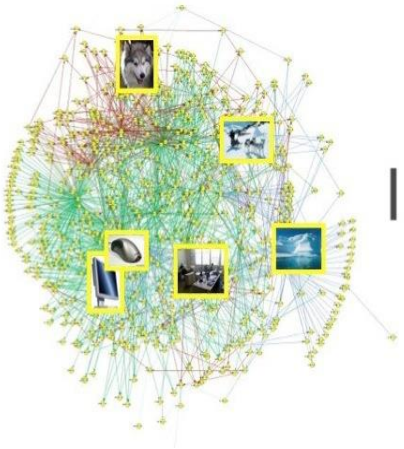


Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

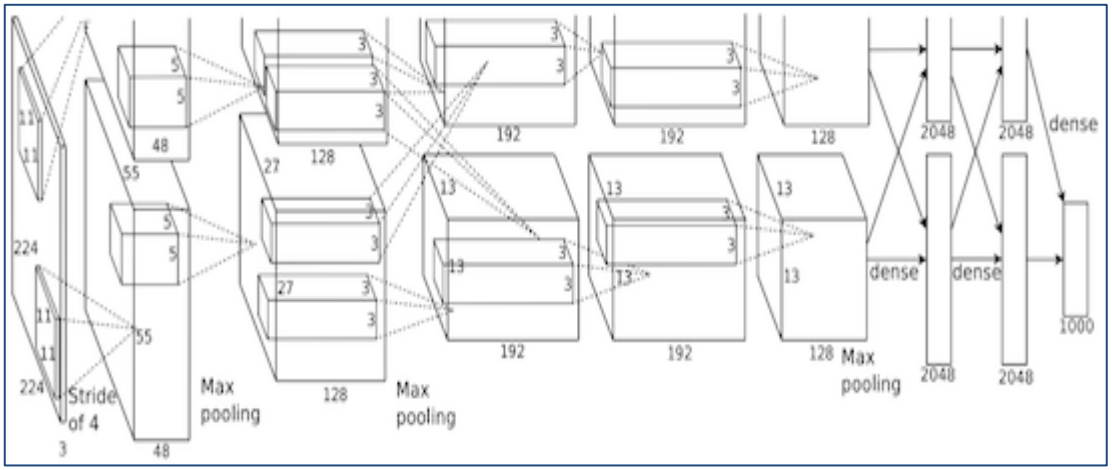
A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]



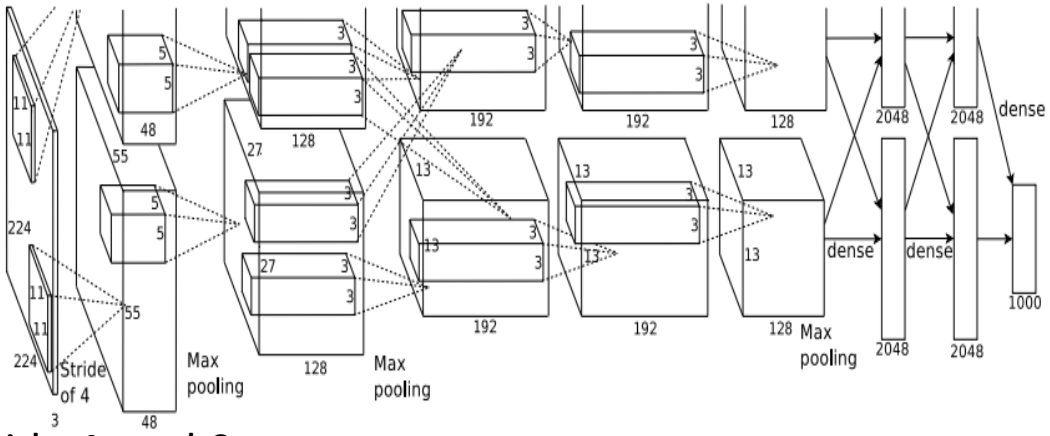
IMAGENET



"AlexNet"

Case Study: AlexNet

[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- **7 CNN ensemble: 18.2% -> 15.4%**

Case Study: VGG-16

- Simonyan & Zisserman won the ImageNet ILSVRC-2014 challenge.
- They published a paper called “Very Deep Convolutional Networks for Large-Scale Image Recognition”
- They compare networks of up to 19 layers.
- They also **published** their network
- It is one of the most popular pre-trained networks and available for a lot of deep learning libraries, including Keras.
- https://keras.io/applications/

VGG-16 Architecture

input: 224x244 RGB image

64 Conv 3x3
64 Conv 3x3
maxpool

128 Conv 3x3
128 Conv 3x3
maxpool

256 conv 3x3
256 conv 3x3
256 conv 3x3
maxpool

512 conv 3x3
512 conv 3x3
512 conv 3x3
maxpool
512 conv 3x3
512 conv 3x3
512 conv 3x3
maxpool
4096 FC
4096 FC
1000 FC
softmax

All layers using ReLU
Trained with SGD and momentum

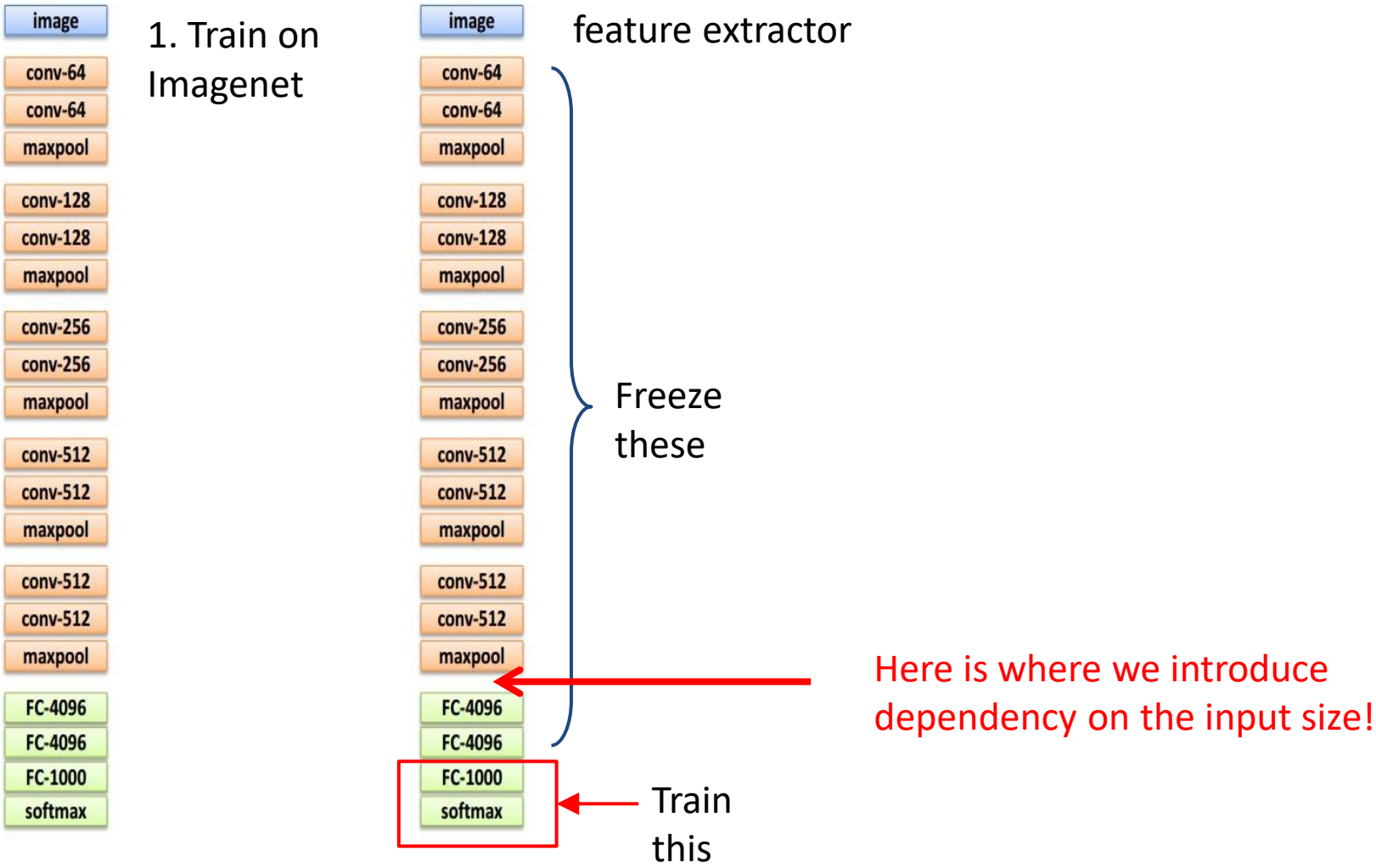
VGG-16 - Training

- SGD + momentum
- With Gaussian initialization training needed tricks:
  - First trained smaller network (less layers)
  - Used the smaller network as initialization for deeper network
- With Glorot initialization it could be trained from scratch!
- See paper for further details: https://arxiv.org/pdf/1409.1556.pdf

Fine Tuning

"You need a lot of a data if you want to train/use CNNs"

Fine Tuning with CNNs



http://cs231n.github.io/

Intermediate Output in Keras

- Simple: Define a new model that only has the first n layers
- Here `model` is the original full network

```
intermediate_layer_model = Model(
      inputs=model.input,
      outputs=model.get_layer(layer_name).out
put)

intermediate_output =
intermediate_layer_model.predict(data)
```

https://keras.io/getting-started/faq/#how-can-i-obtain-the-output-of-an-intermediate-layer

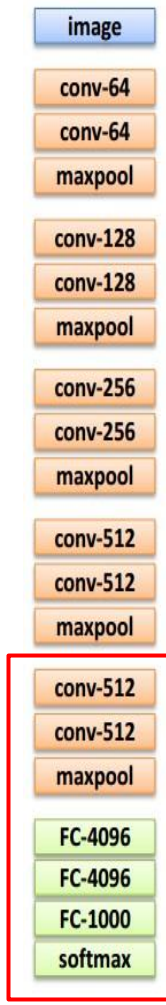
Fine Tuning with CNNs

1. Train on Imagenet

2. Small dataset: **feature extractor**

Freeze these

Train this



3. Medium dataset: **finetuning**

more data = retrain more of the network (or all of it)

Freeze these

tip: use only ~1/10th of the original learning rate in finetuning top layer, and ~1/100th on intermediate layers

Train this

http://cs231n.github.io/

Freezing Layers in Keras

```
# set the first 25 layers
# to non-trainable
# weights will not be updated

for layer in model.layers[:25]:
    layer.trainable = False
```

After freezing the layers you need to compile the model again with the optimizer of your choice and a very low learning rate

	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You're in trouble…
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

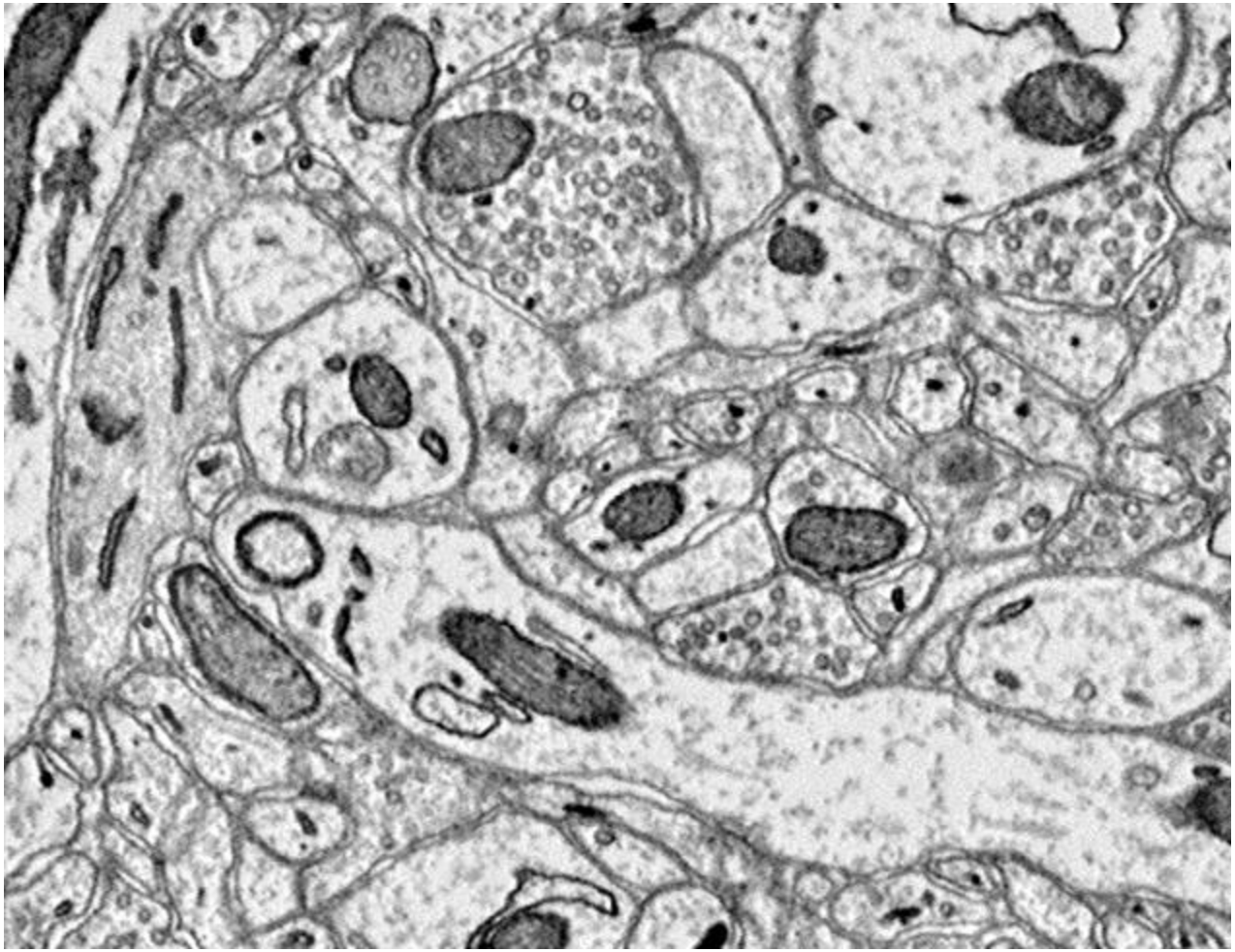
http://cs231n.github.io/

Unsupervised Pretraining

- In 2006 deep learning had a breakthrough
- Unsupervised Pretraining
- Using Autoencoders or RBMs
- Leverage unlabeled data
- Learn to reproduce signal
- Use labeled data to fine tune

- This is what humans do all the time!

Pretrained Visual System

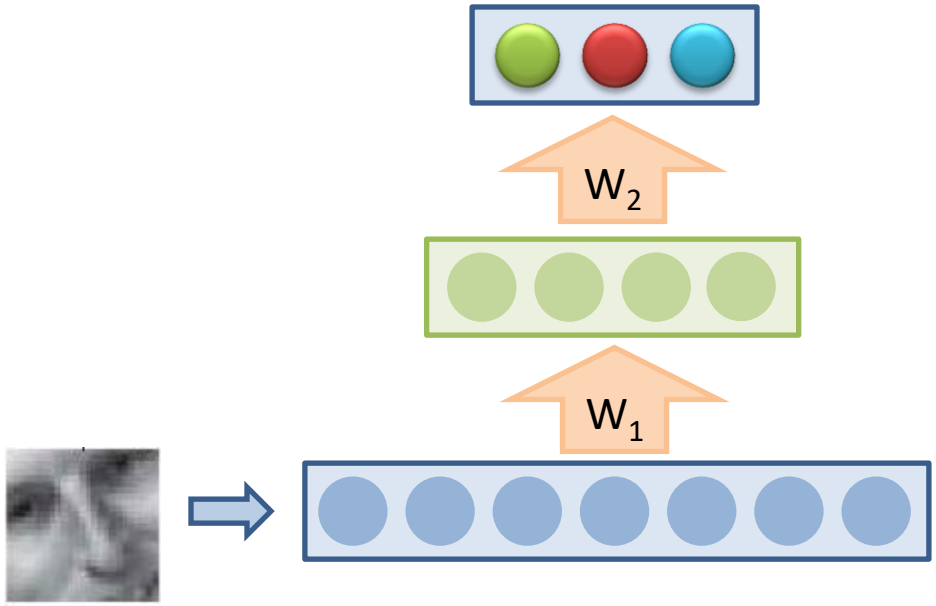


Autoencoder

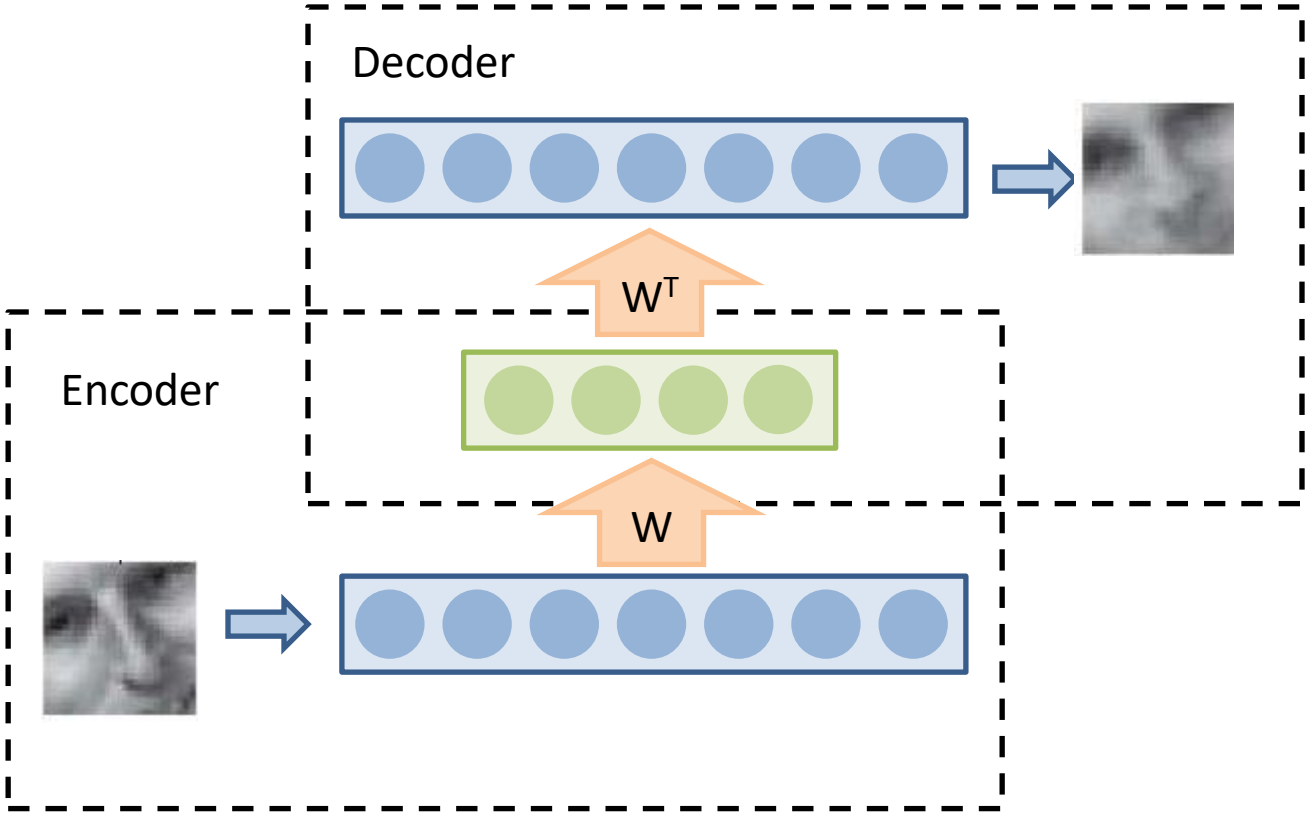
- This is what Google used for their Google brain
- Basically just a MLP
- Output size is equal to input size

- Popular for pre-training a network on unlabeled data

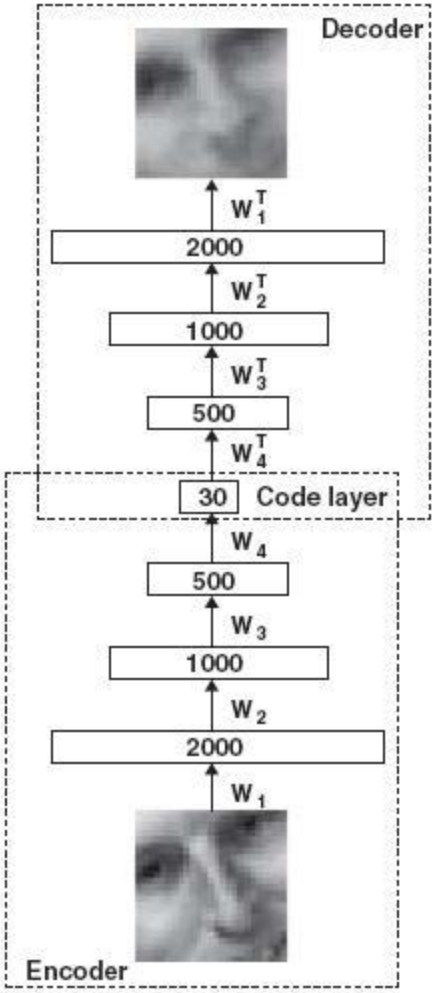
MLP



Autoencoder



Deep Autoencoder

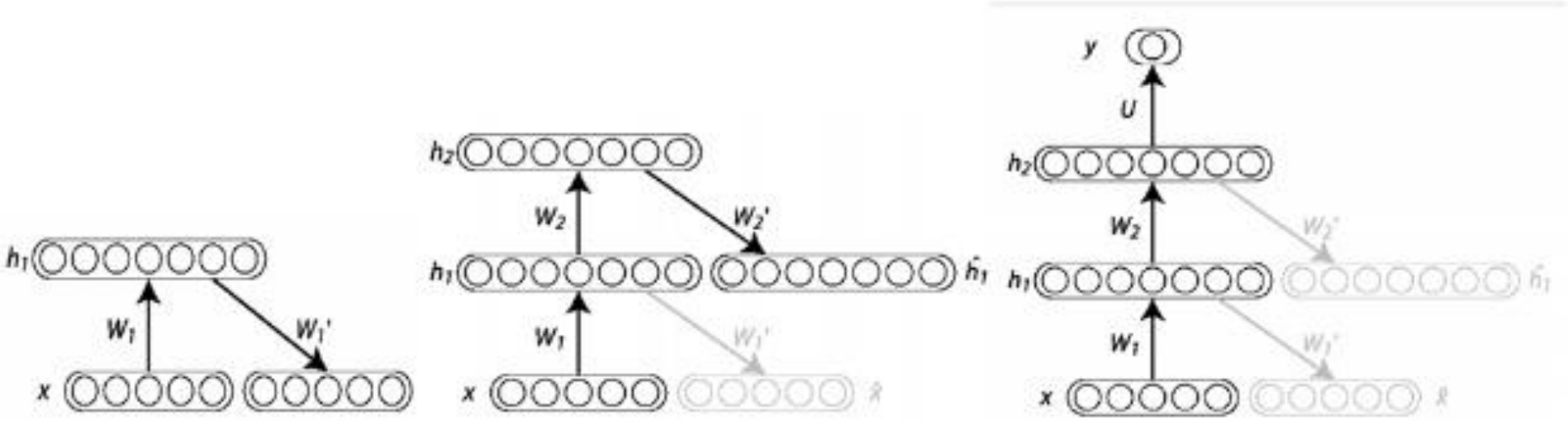


- Reconstruct image from learned low dimensional code
- Weights are tied
- Learned features are often useful for classification
- Can add noise to input image to prevent overfitting

Salakhutdinov & Hinton, NIPS 2007

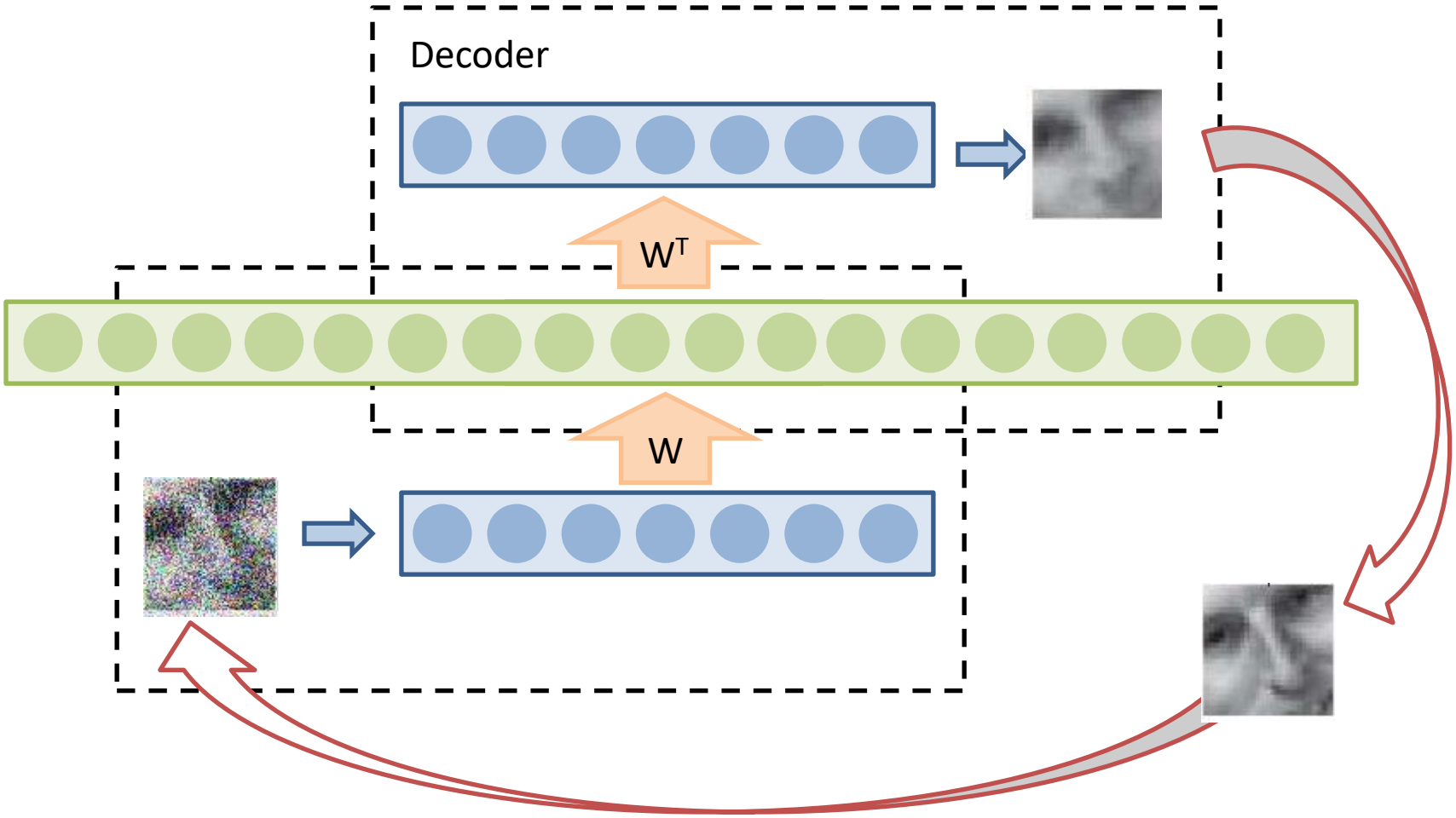
Pretraining: Stacked Denoising Auto-encoder

- Stacking Auto-Encoders



from: Bengio ICML 2009

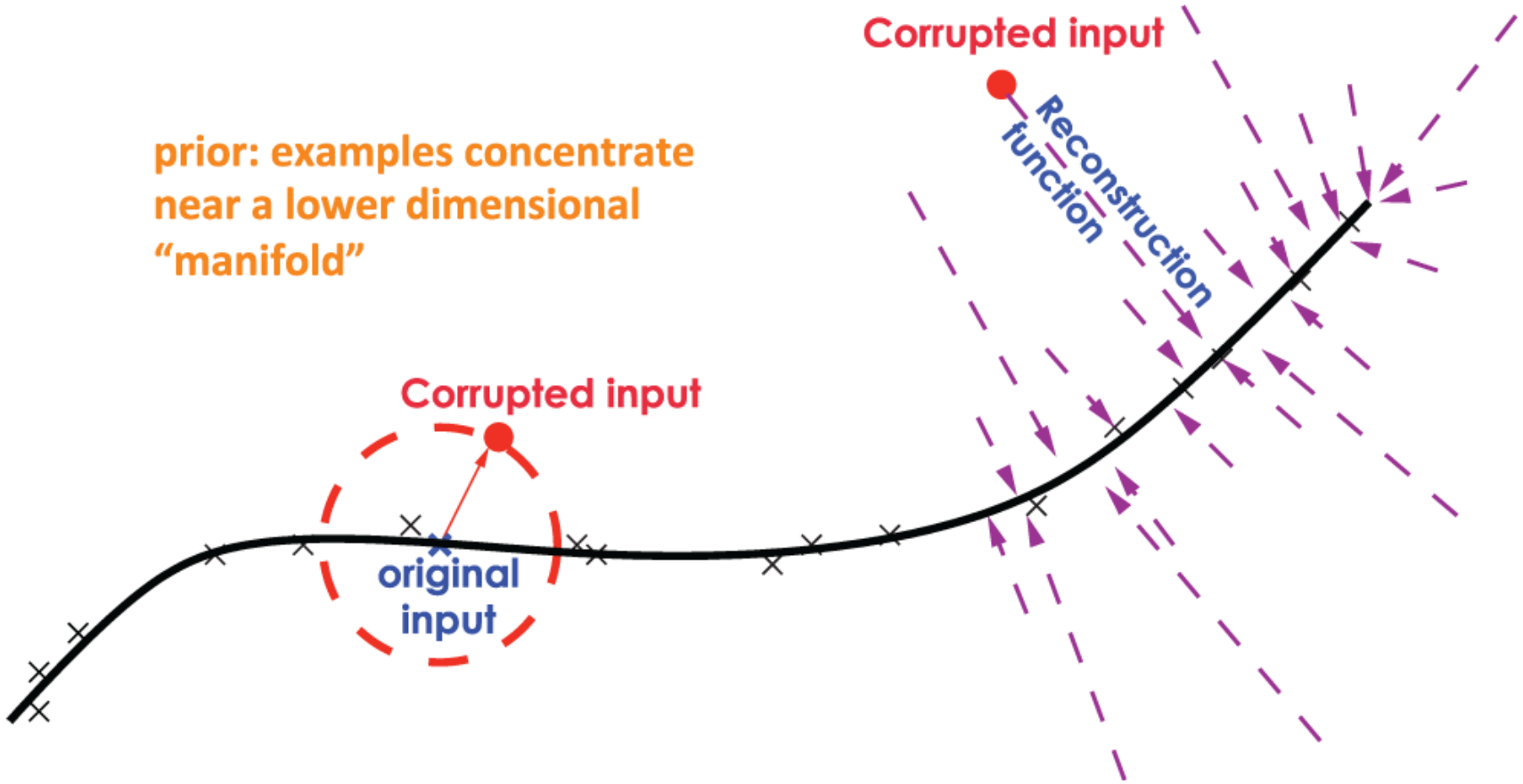
Denoising Autoencoder



Denoising Autoencoder - Examples

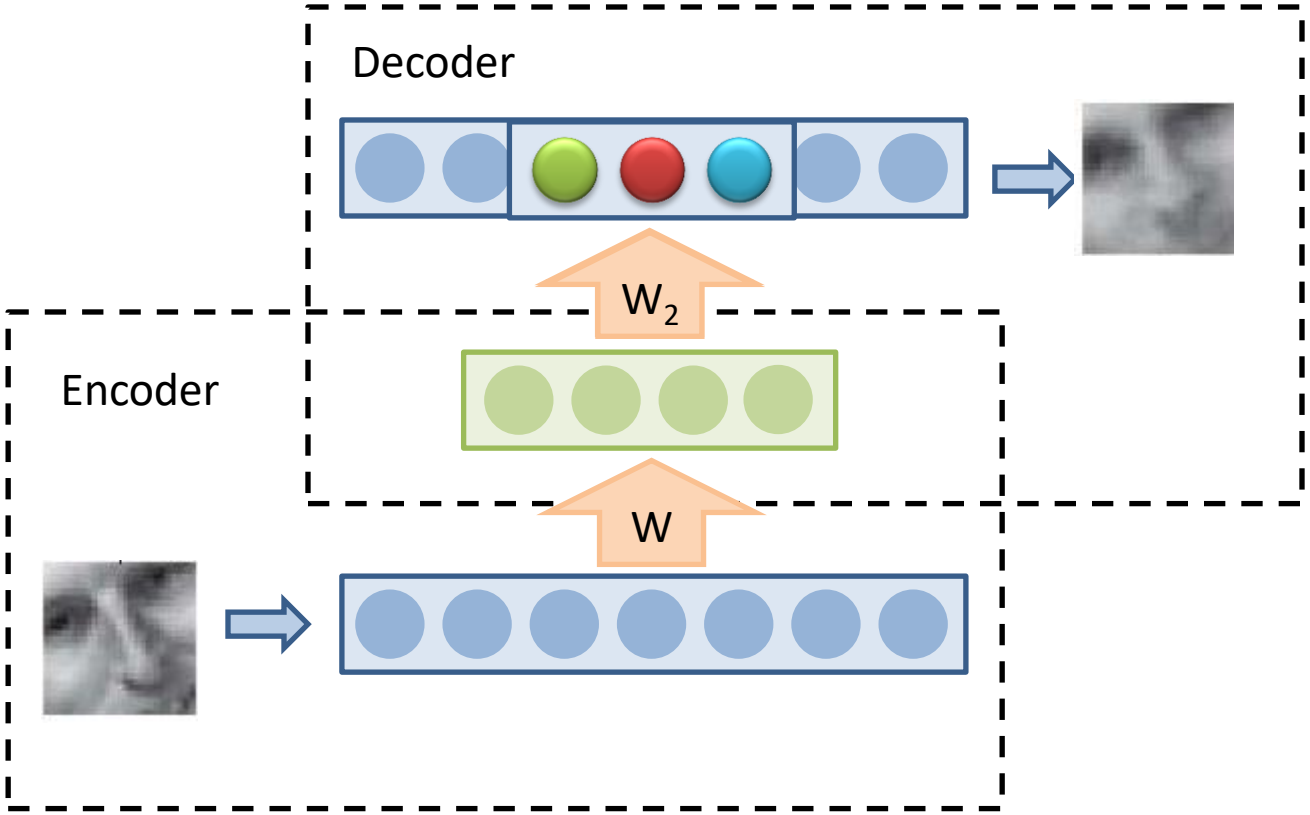
https://blog.keras.io/building-autoencoders-in-keras.html

Denoising Autoencoder

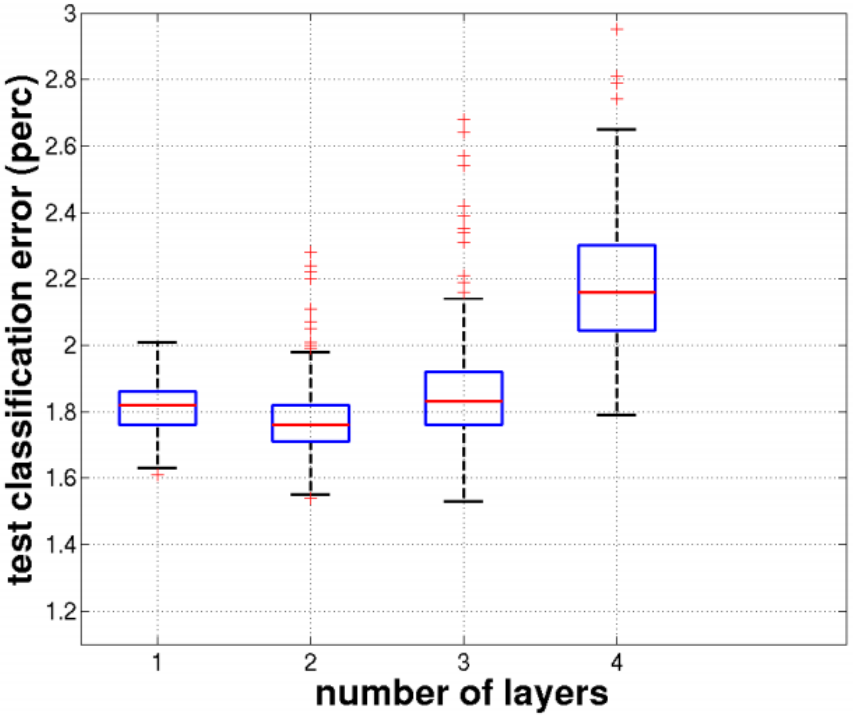


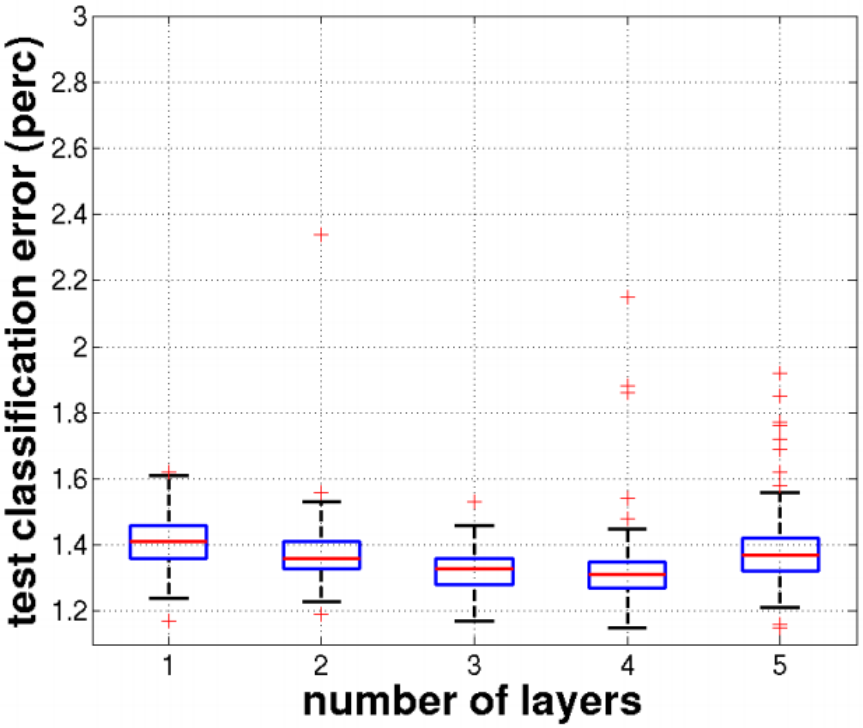
Yoshua Bengio

Unsupervised Pretraining



Unsupervised Pretraining

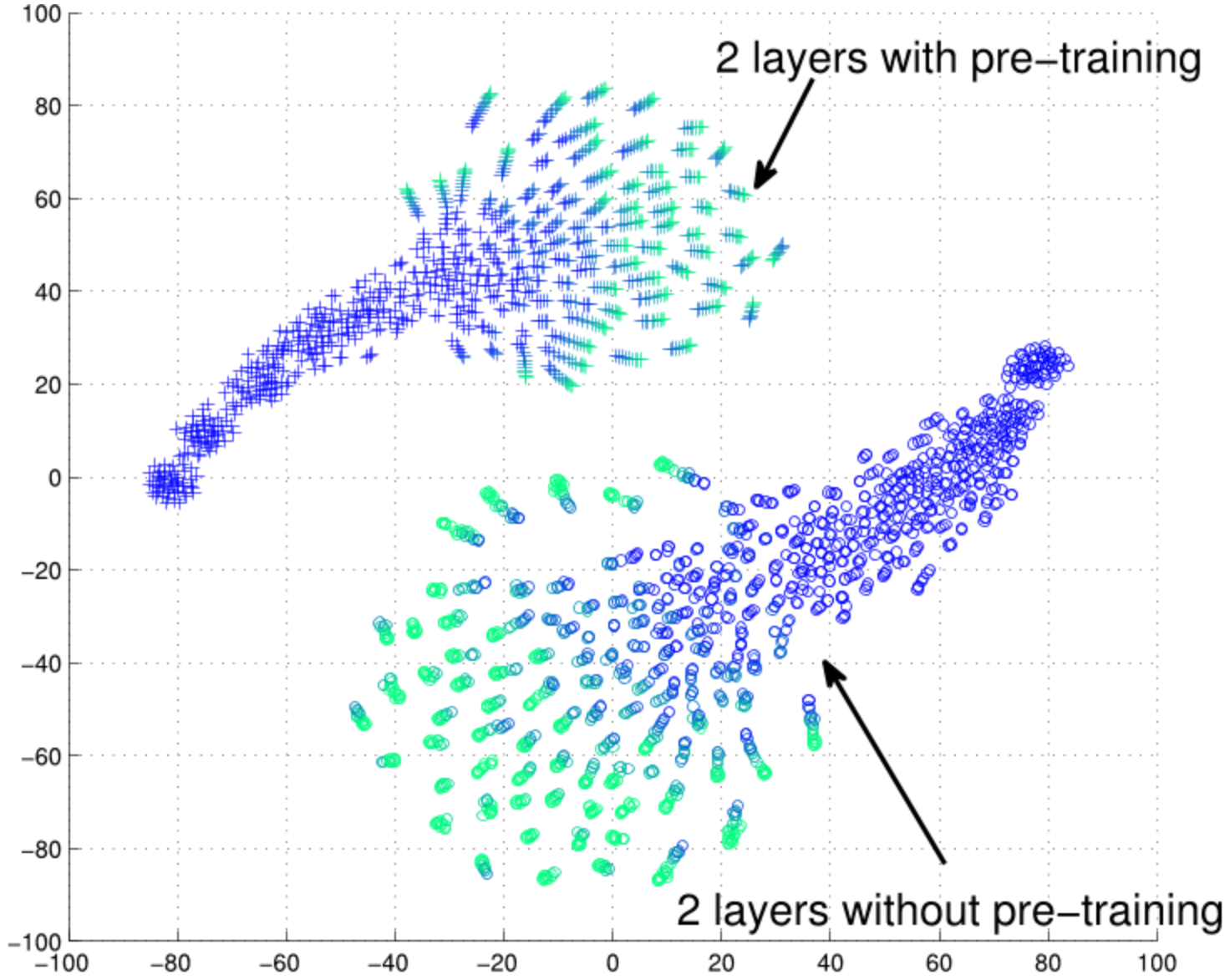




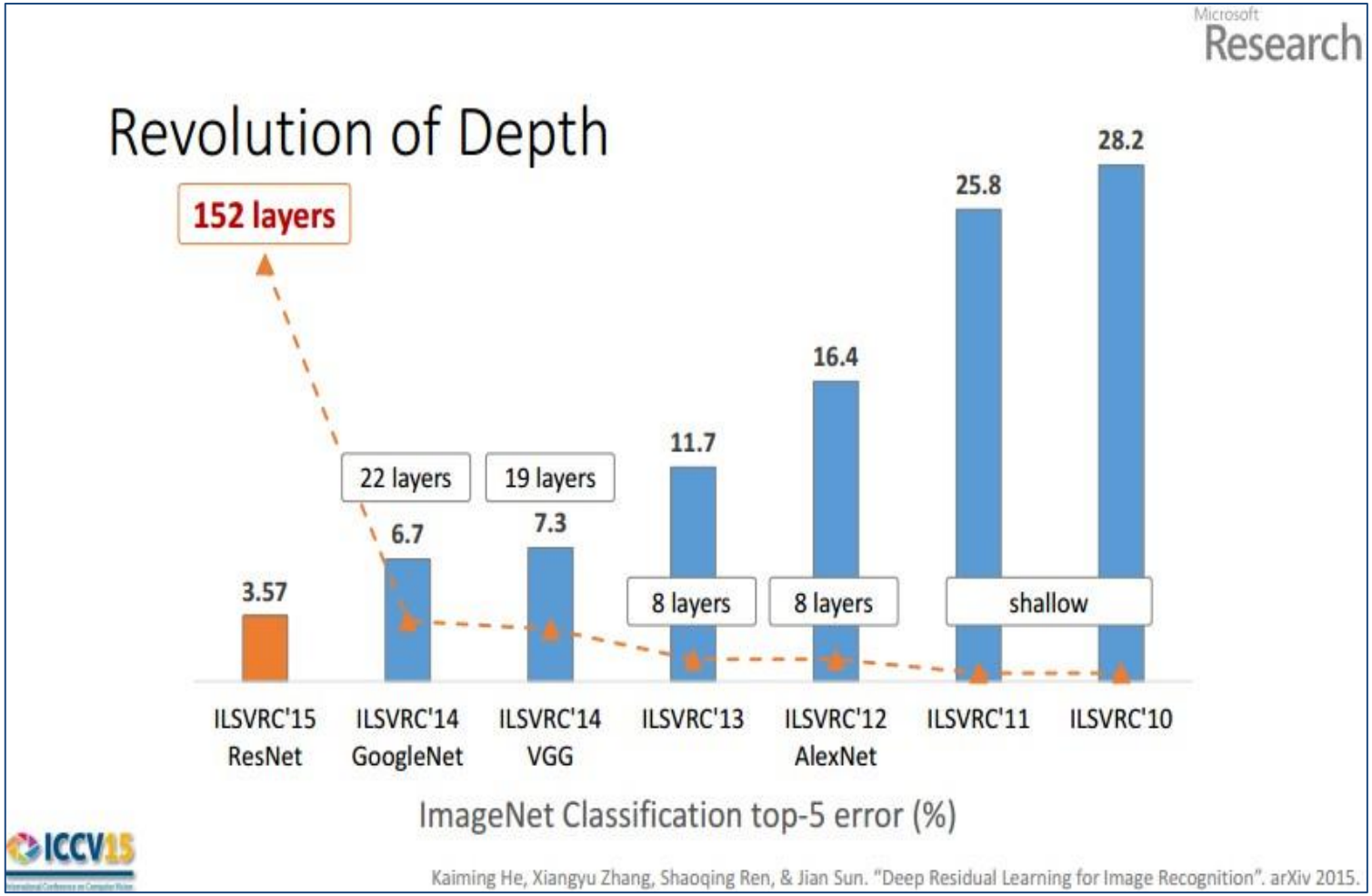
without pretraining

with pretraining

Erhan et al. “Why Does Unsupervised Pre-training Help Deep Learning?”, Journal of Machine Learning Research 11 (2010) 625-660



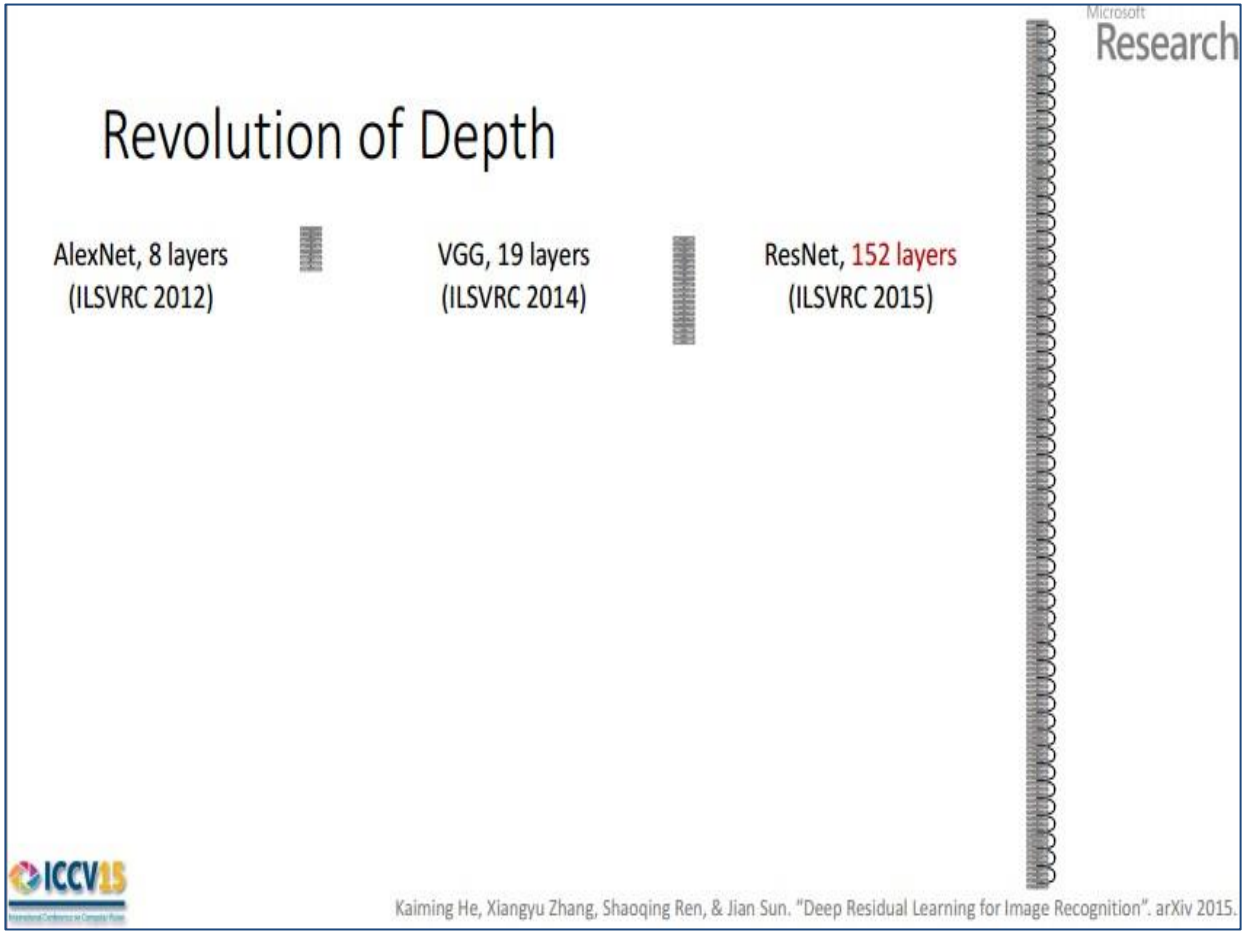
Erhan et al. “Why Does Unsupervised Pre-training Help Deep Learning?”, Journal of Machine Learning Research 11 (2010) 625-660



(slide from Kaiming He's recent presentation)

Case Study: ResNet *[He et al., 2015]*

ILSVRC 2015 winner (3.6% top 5 error)

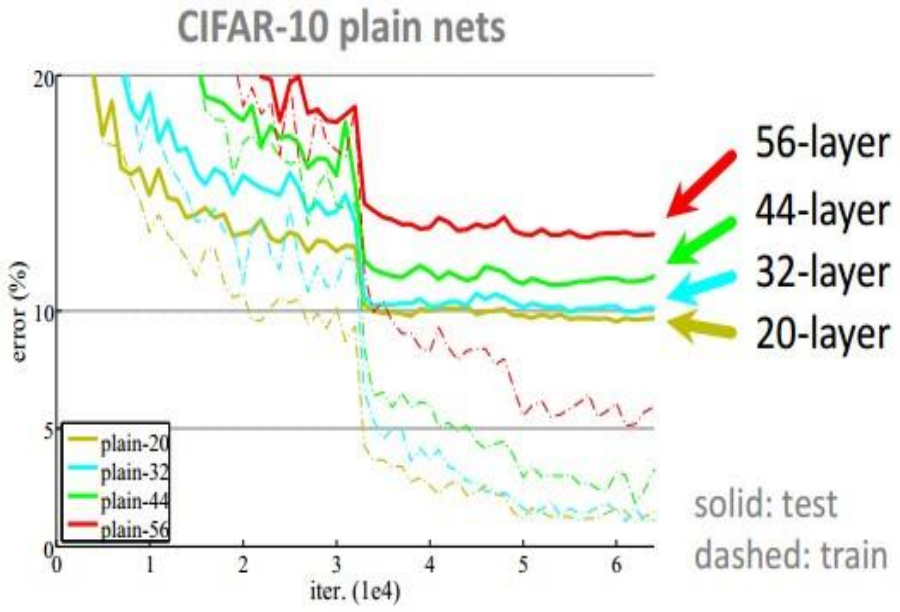


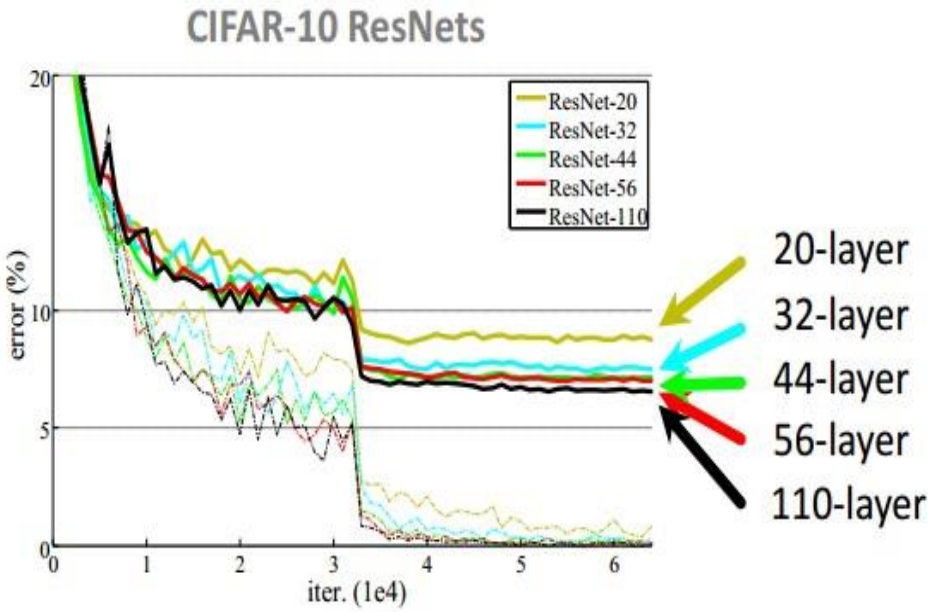
(slide from Kaiming He's recent presentation)

2-3 weeks of training on 8 GPU machine

at runtime: faster than a VGGNet! (even though it has 8x more layers)

CIFAR-10 experiments

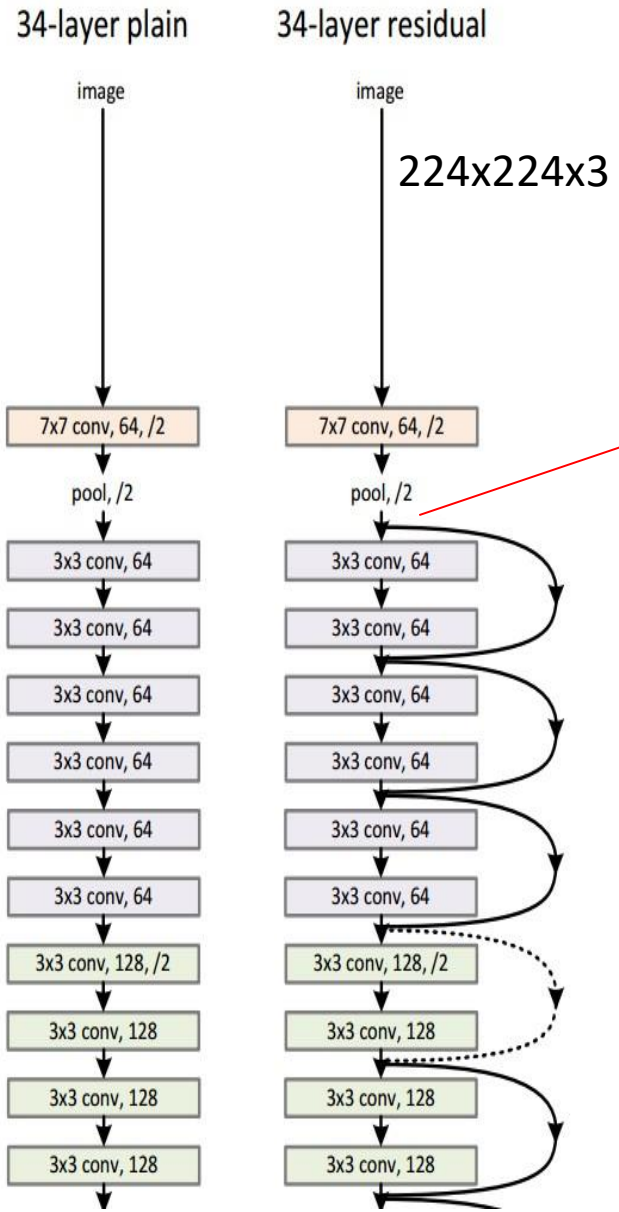




http://cs231n.github.io/

Case Study: ResNet

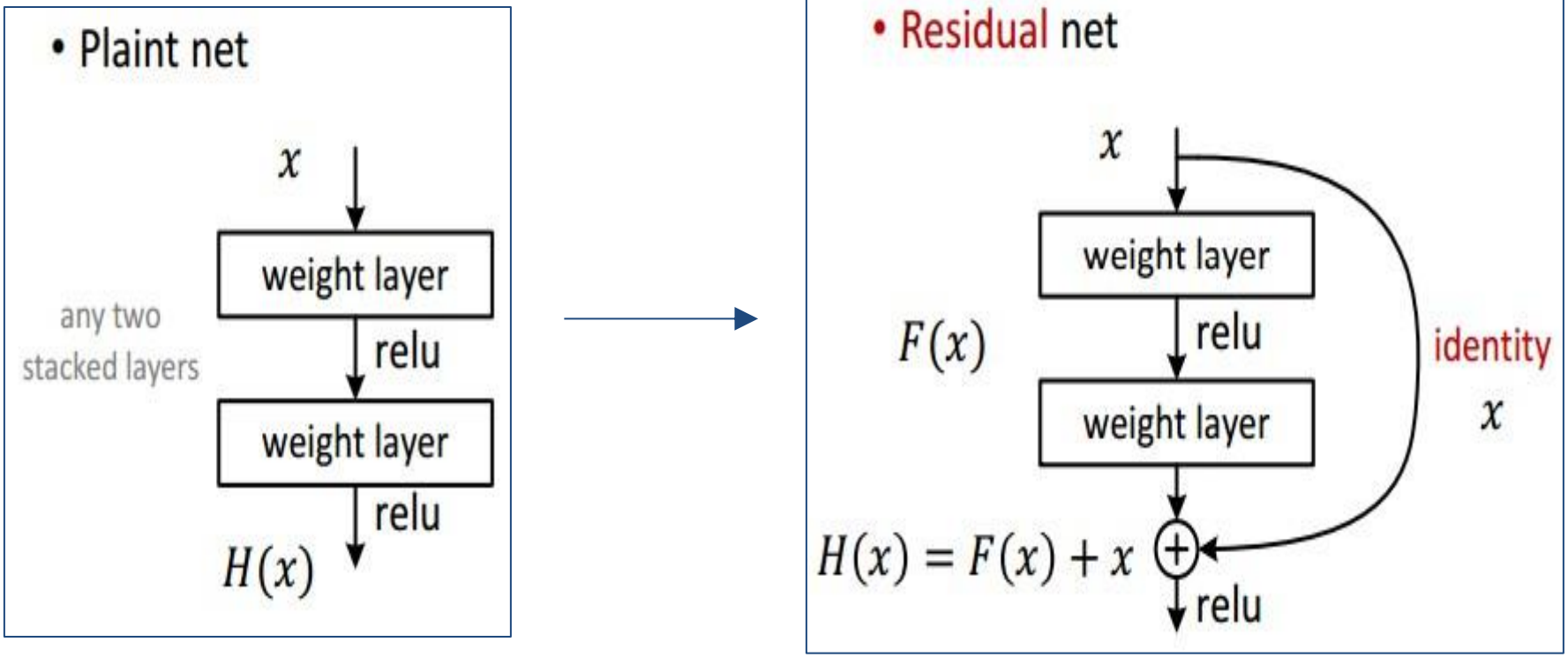
[He et al., 2015]



spatial dimension only 56x56!

Case Study: ResNet

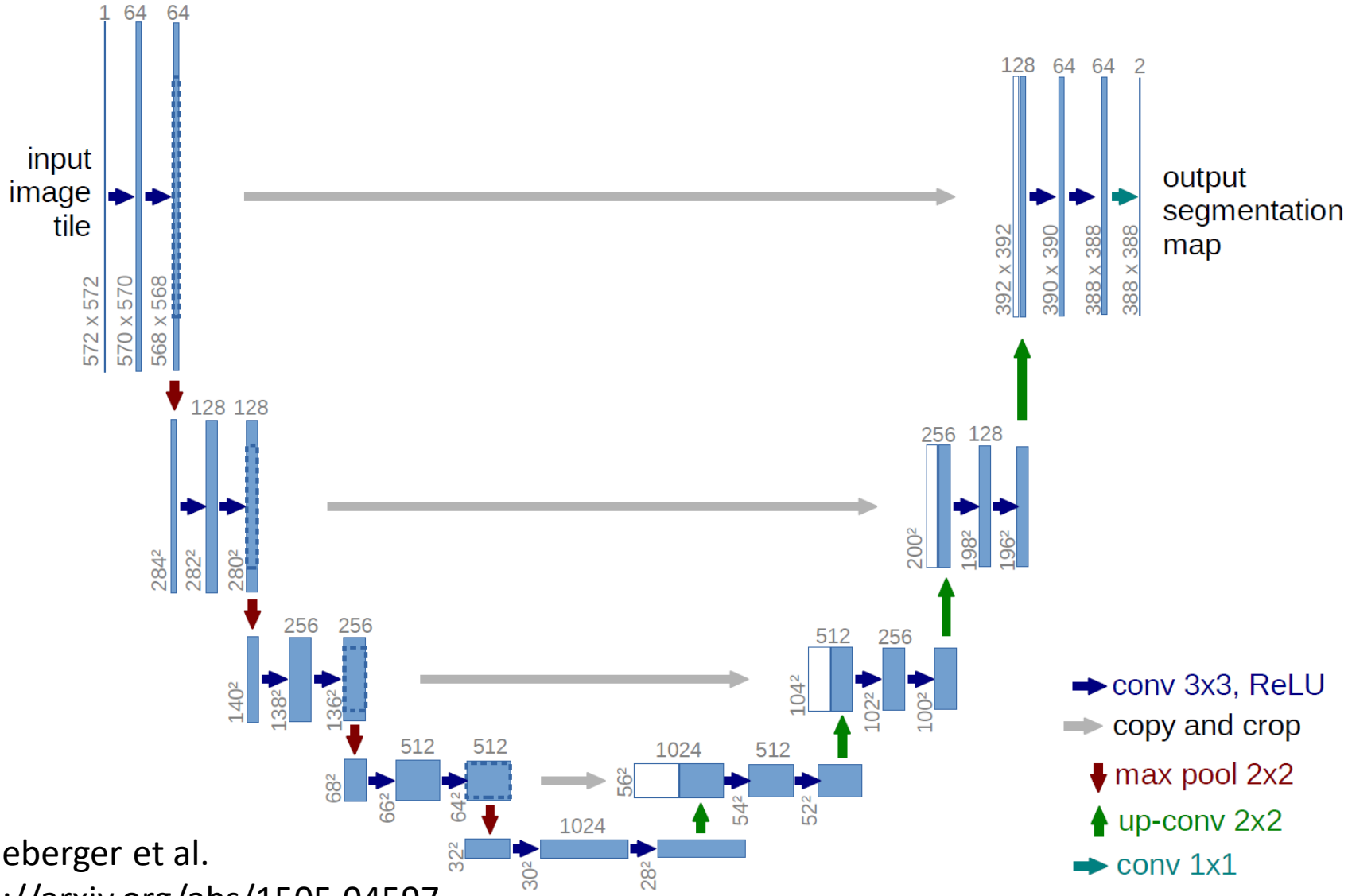
[He et al., 2015]



Case Study: ResNet *[He et al., 2015]*

- Batch Normalization after every CONV layer
- Xavier/2 initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

U-Net



Ronneberger et al.
https://arxiv.org/abs/1505.04597

Build Ensembles

- Train multiple independent models
- At test time average their results

Enjoy 2% extra performance

- Can also get a small boost from averaging multiple model checkpoints of a single model.

Lessons Learned from Random Forest

- We want to make the models as different as possible to minimize the correlations between their errors.
- We can use different initializations or different architectures or different subsets of the training data.
- It is helpful to over-fit the individual models.

http://www.ttic.edu/dl/dark14.pdf

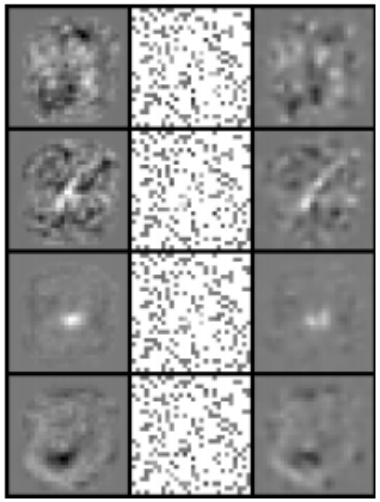
Ensembles Are Inefficient

- A big ensemble is highly redundant. It has very little knowledge per parameter.
- At test time we want to minimize the amount of computation and the memory footprint.
- These constraints are generally much more severe at test time than during training.

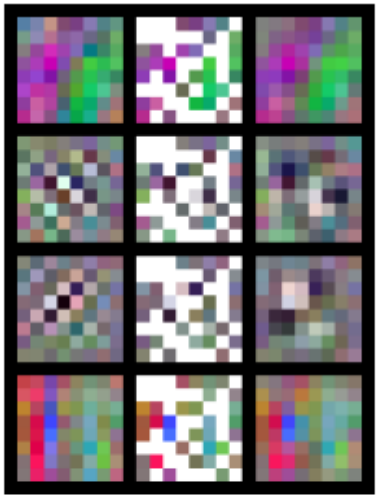
http://www.ttic.edu/dl/dark14.pdf

Large Networks are Redundant

- *Work by Misha Denil* et al.
- Only look at a fraction of activations and predict the rest



MLP on MNIST



CNN on CIFAR-10

https://arxiv.org/pdf/1306.0543.pdf

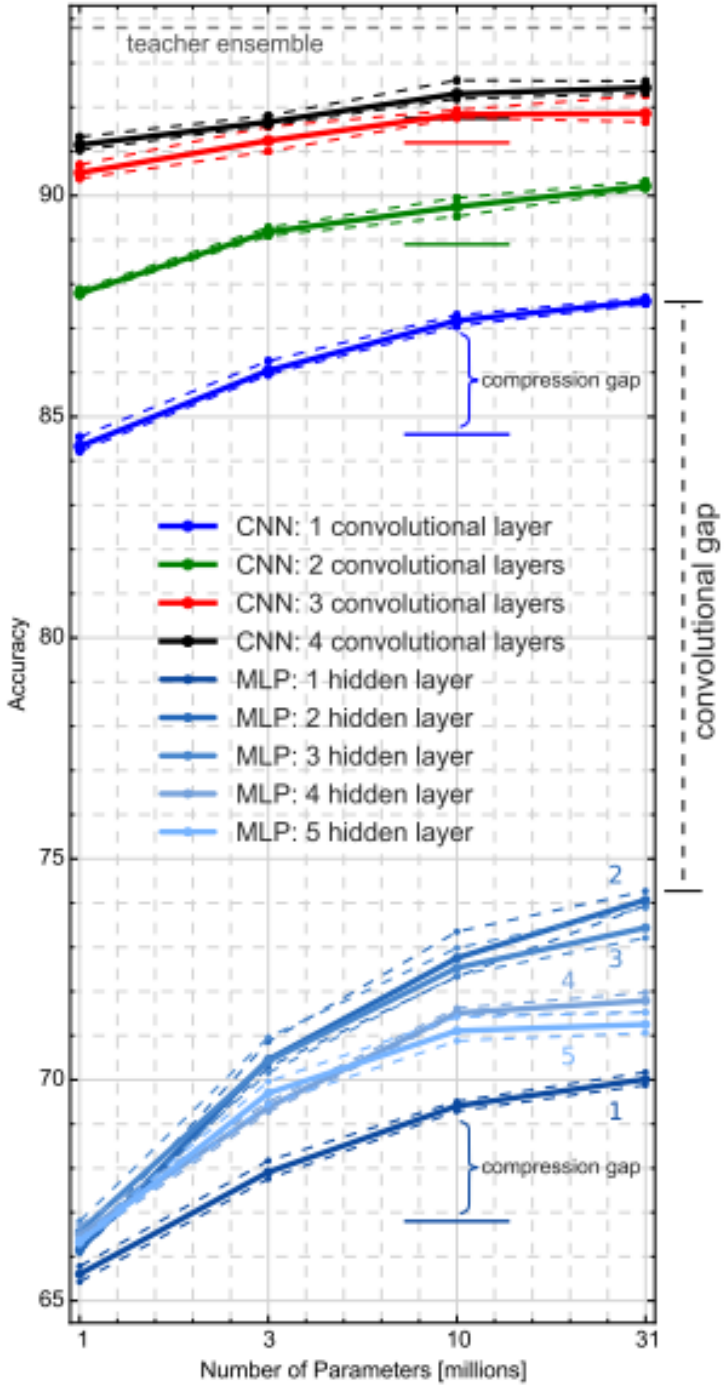
Knowledge Distillation

- Hinton refers to this as using dark knowledge
- Student is trained on the scores of the teacher, not the hard classification

Teacher: Big ensemble, large network, slow at test time

Student: Small and efficient network.

- Do we need networks to be deep networks?
- Work by G. Urban et al.
- Teacher:
  - 16 CNN ensemble
  - Each 8 conv and 2 fc



https://arxiv.org/abs/1603.05691

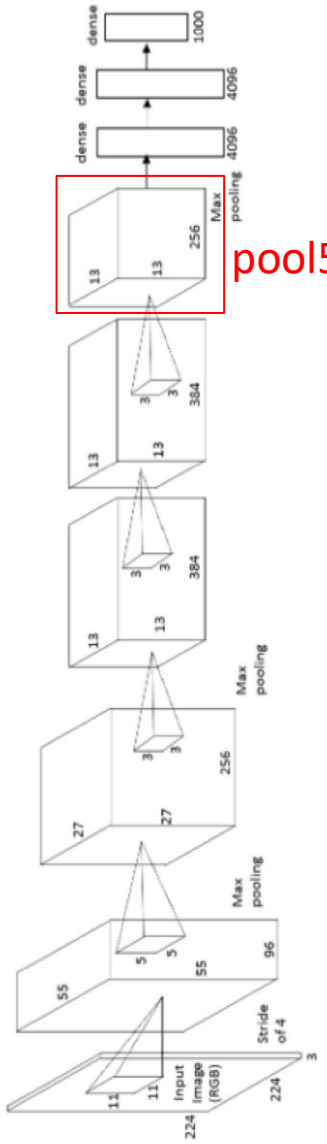
Visualizing Network Internals

- Deep networks are this gigantic black box
- It would be really neat to be able to see what is going non inside
- Let’s look at some ideas

Visualize patches that maximally activate neurons

one-stream AlexNet

Figure 4: Top regions for six pool$_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

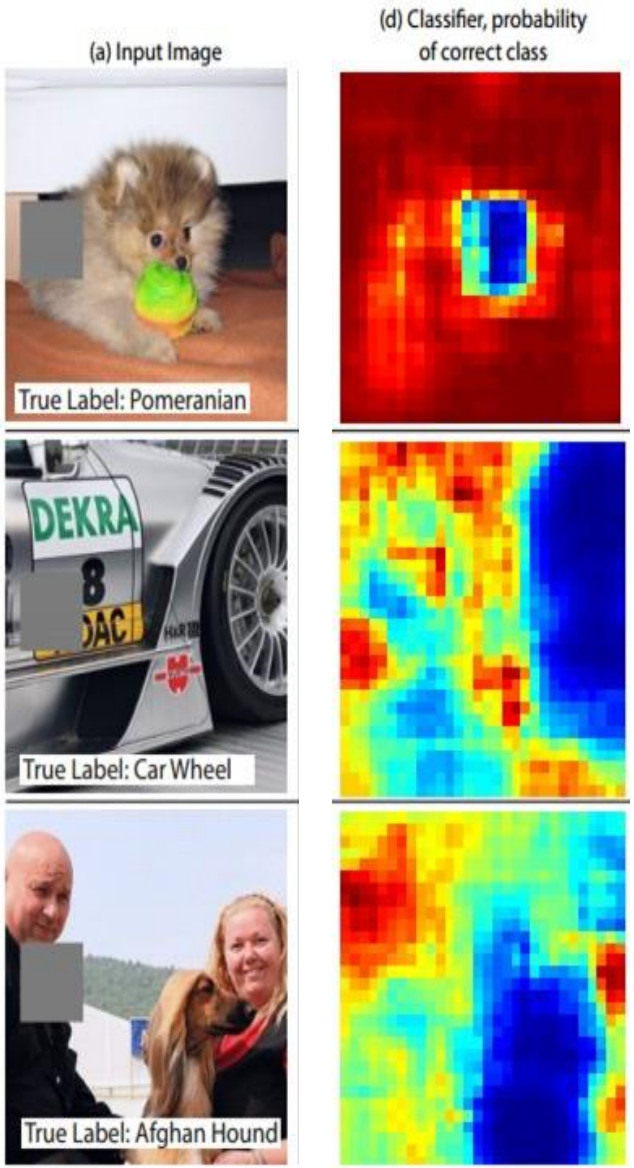


Rich feature hierarchies for accurate object detection and semantic segmentation
[Girshick, Donahue, Darrell, Malik]

http://cs231n.github.io/

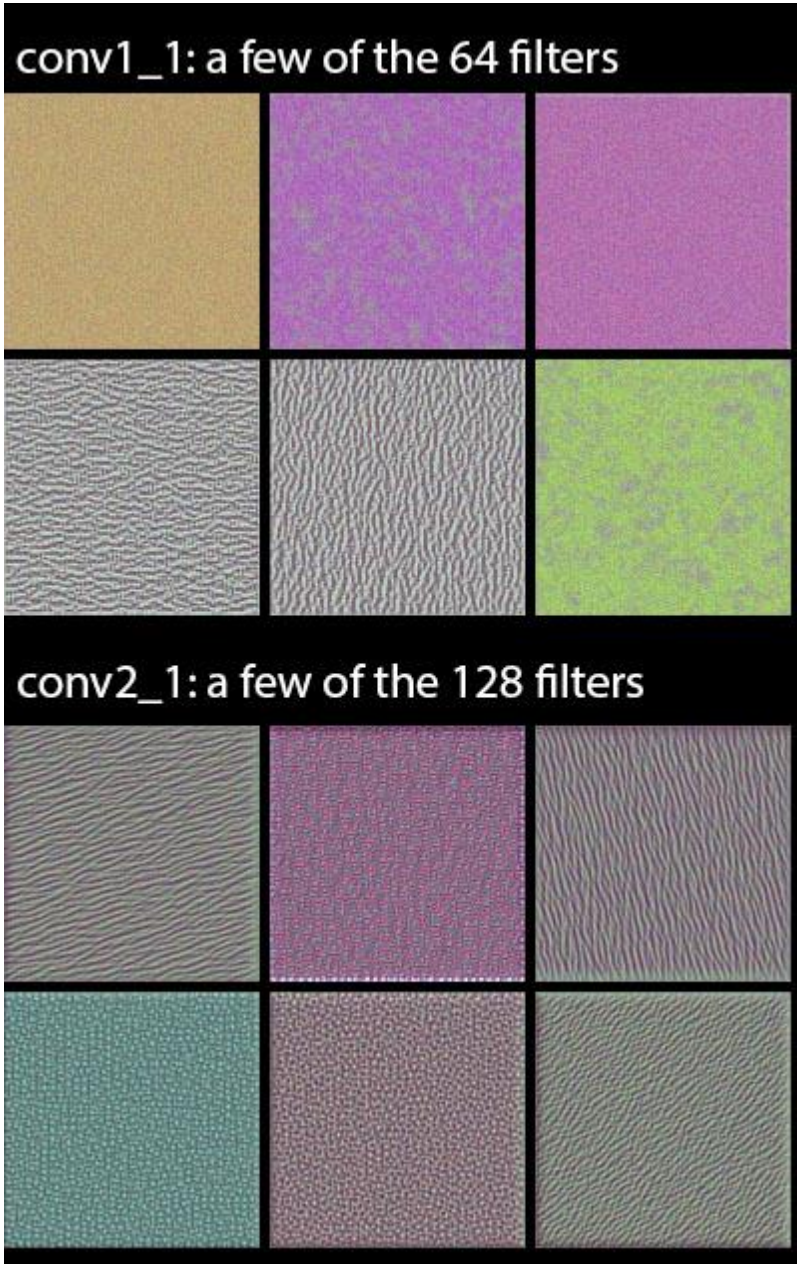
Occlusion experiments

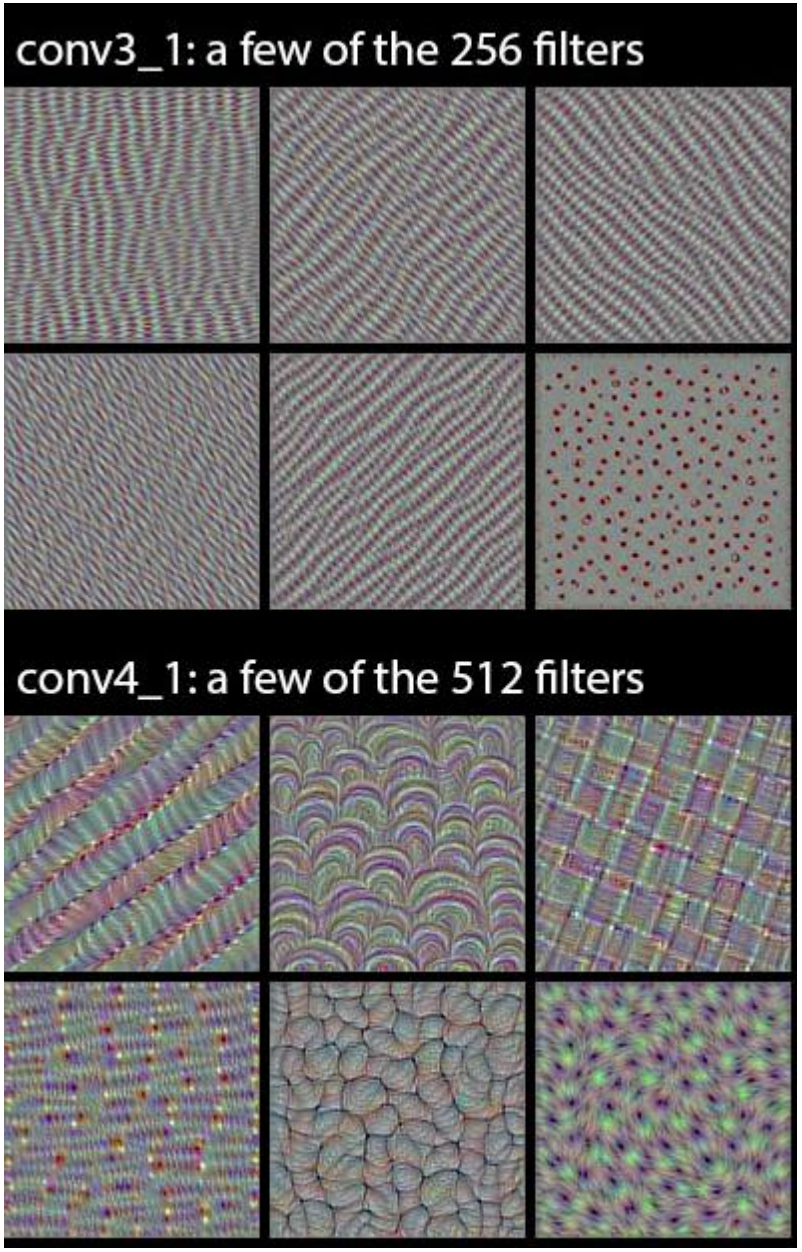
[Zeiler & Fergus 2013]



(as a function of the position of the square of zeros in the original image)

http://cs231n.github.io/





https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

The Optimization Problem

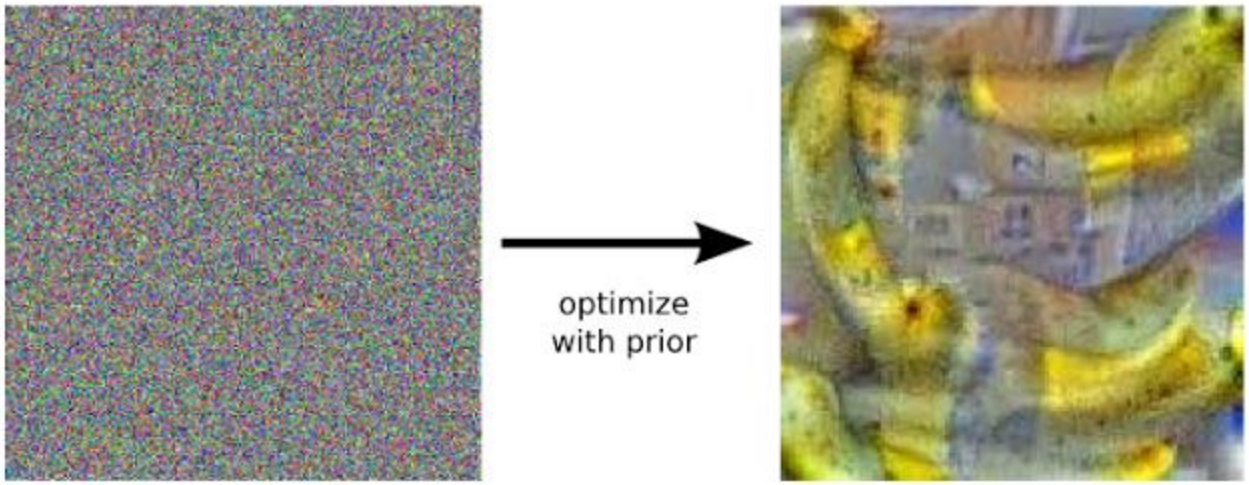
- score = mean(activation(filter, image))
- Compute gradient
- Use gradient ascent to optimize input image

- Needs a bit of gradient normalization
- Just a few lines of code

https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

Google Version of This Vis

- Similar idea
- Added a smoothness term to make input more image like

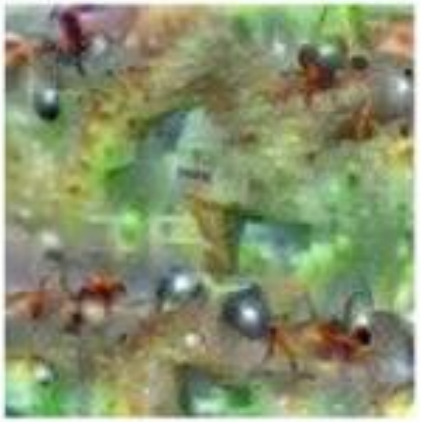


https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Results are Beautiful

Hartebeest

Measuring Cup



Ant

Starfish

Anemone Fish

Banana

Parachute

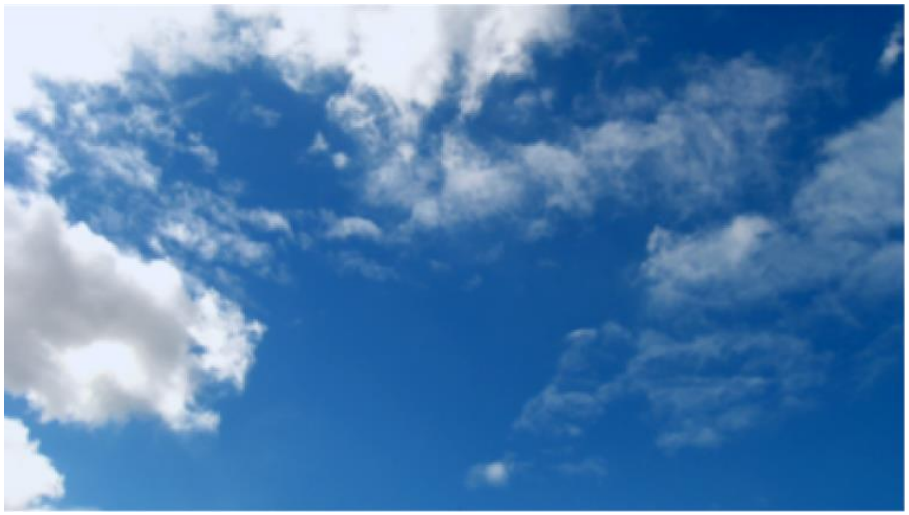
Screw

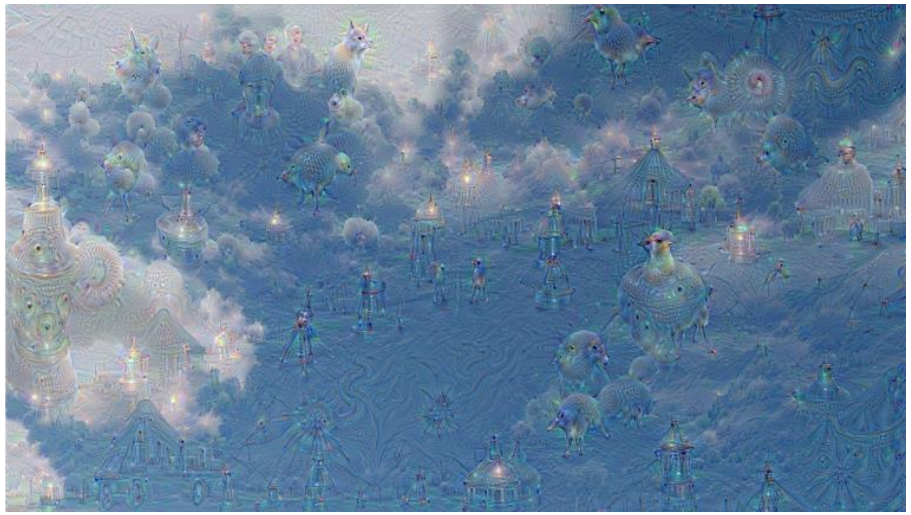
https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

And Funny

https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

What If We Don’t Start with Noise?





"Admiral Dog!"

"The Pig-Snail"

"The Camel-Bird"

"The Dog-Fish"

https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Deep Dreams

- https://www.youtube.com/watch?v=SCE-QeDfXtA&t=49s

But what does that tell us?

- We can see the patterns the network learned
- But, Google had to add a prior to make them look like images
- We are still far from AI

- Networks have some intriguing properties

Intriguing properties of neural networks

- We can pose an optimization over the input image to maximize any class score.
- That seems useful.

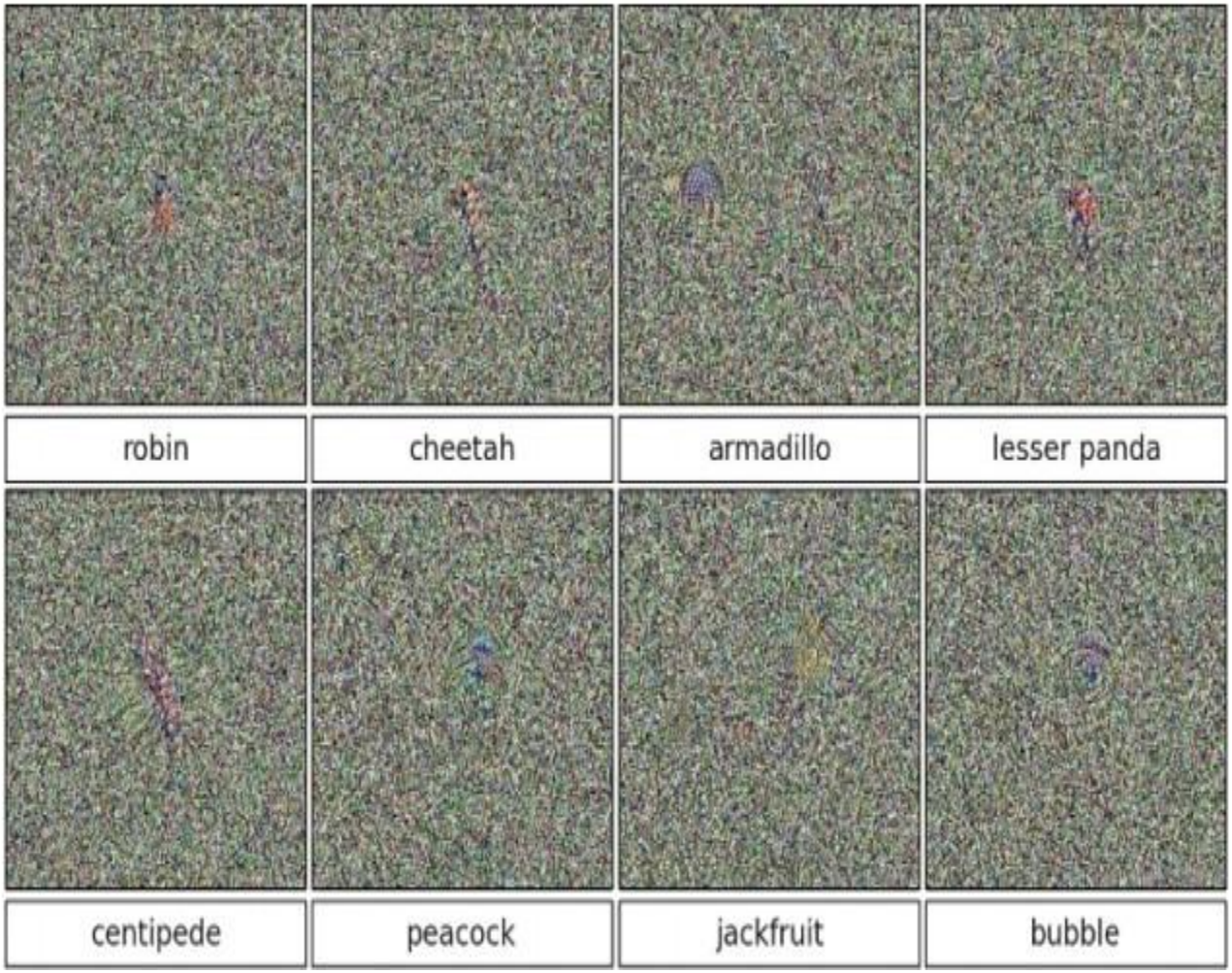
- Can we use this to “fool” ConvNets?

[Intriguing properties of neural networks, Szegedy et al., 2013]

http://cs231n.github.io/

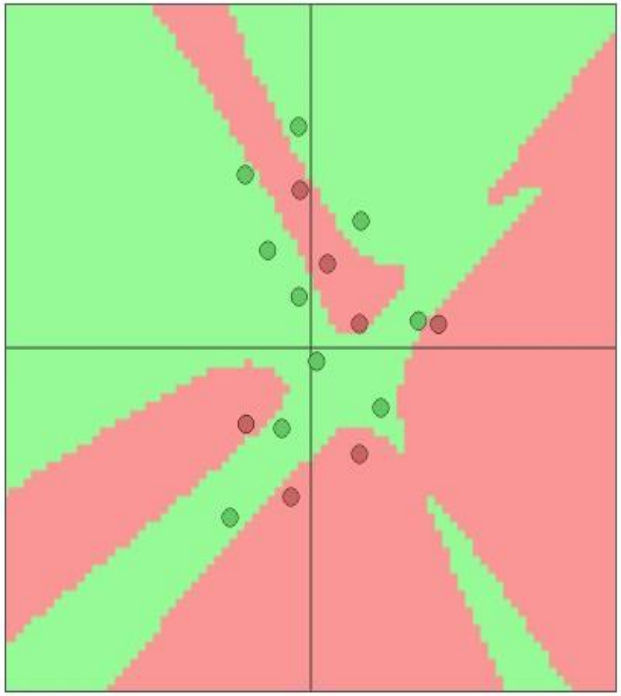
[Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images
Nguyen, Yosinski. Clune. 2014]

>99.6% confidences



EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES
[Goodfellow, Shlens & Szegedy, 2014]

"primary cause of neural networks' vulnerability to adversarial perturbation is their **linear nature**"



Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1

← input example

← weights

class 1 score before:

-2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3

=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1
adversarial x	?	?	?	?	?	?	?	?	?	?

← input example

← weights

class 1 score before:

-2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3

=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1
adversarial x	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5

class 1 score before:

-2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3

=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

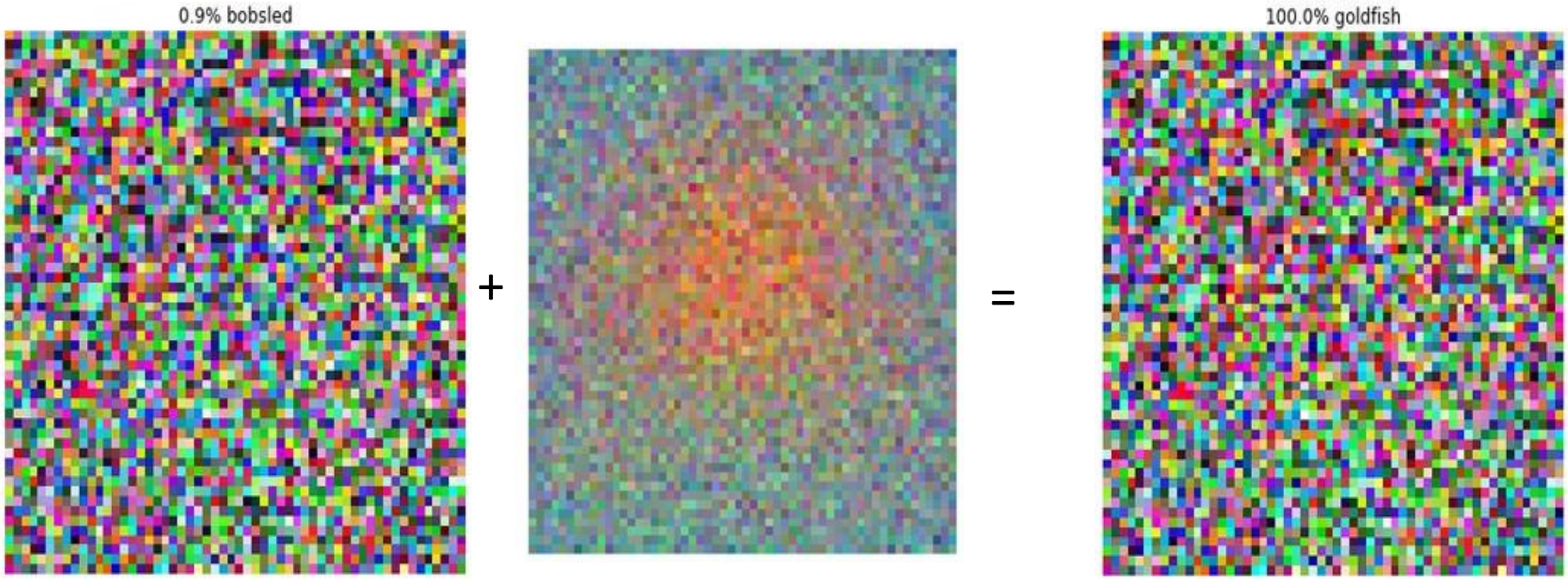
-1.5+1.5+3.5+2.5+2.5-1.5+1.5-3.5-4.5+1.5 = 2

=> probability of class 1 is now 1/(1+e^(-(2))) = 0.88

i.e. we improved the class 1 probability from 5% to 88%

Same Principle

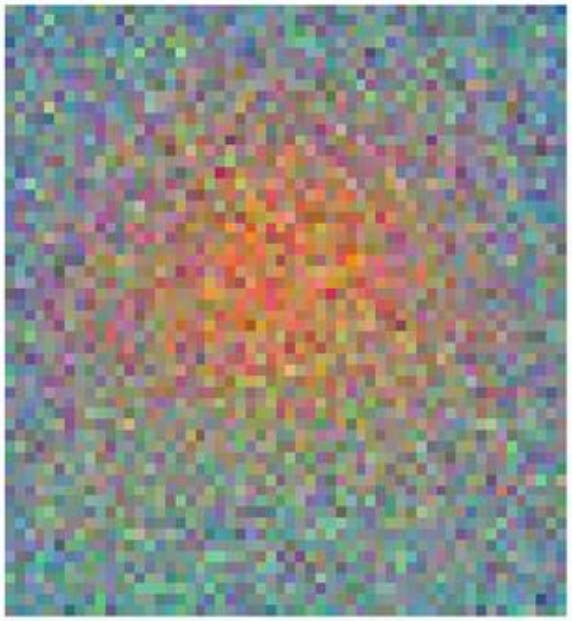
- Lots of tiny variations add up



100% Goldfish

http://cs231n.github.io/

Applied to Image



http://cs231n.github.io/

EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES
[Goodfellow, Shlens & Szegedy, 2014]

"primary cause of neural networks' vulnerability to adversarial perturbation is their **linear nature**"
(and very high-dimensional, sparsely-populated input spaces)

In particular, this is not a problem with Deep Learning, and has little to do with ConvNets specifically. Same issue would come up with Neural Nets in any other modalities.

Summary

- Randomization can be better than grid search
- Deeper is better
- Special connections enable VERY deep networks
- Fine tuning enables deep learning for smaller data sets
- Visualizations can help to understand what was learned
- Deep networks are (once again) not as clever as we think