ECE499/ECE590
Machine Learning for Embedded Systems (Fall 2021)

Lecture 3: Deep Convolutional Neural Networks (CNN)

Weiwen Jiang, Ph.D.

Electrical and Computer Engineering

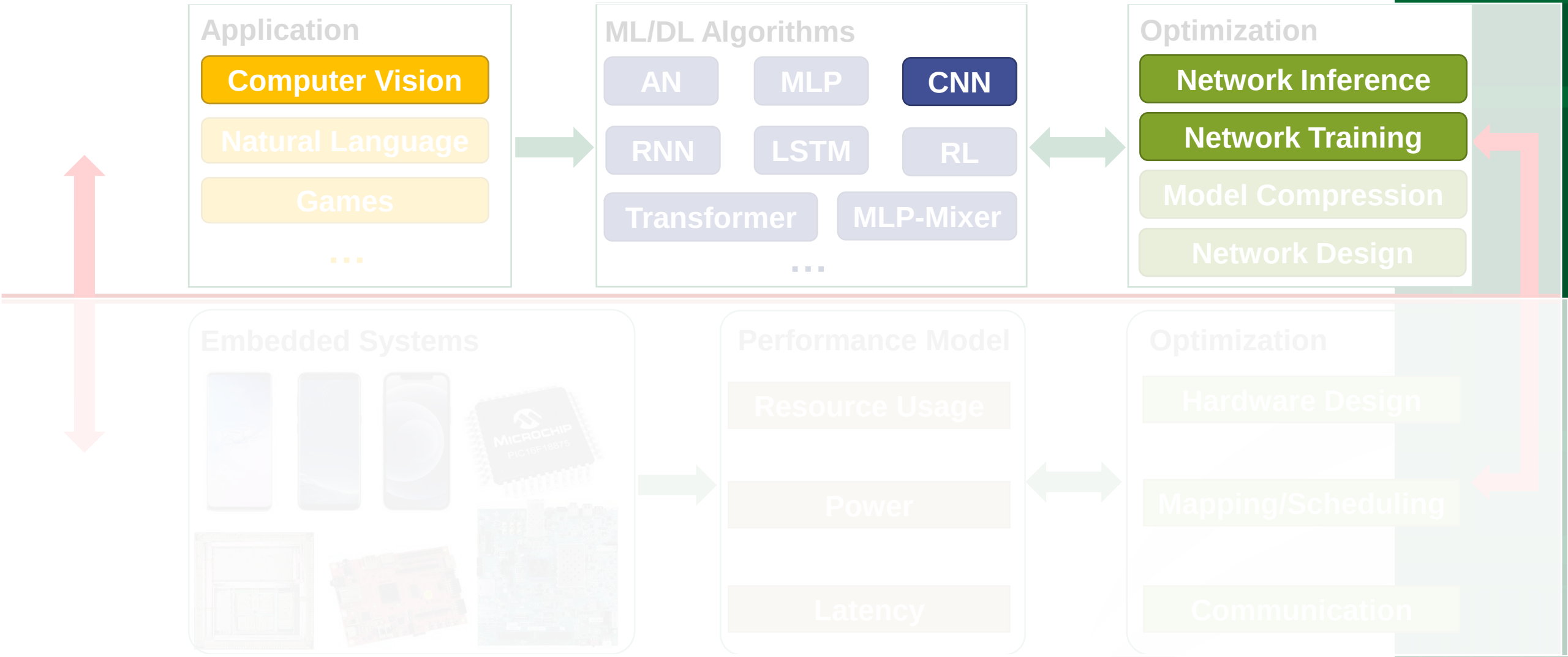
George Mason University

wjiang8@gmu.edu

Artificial Neuron Design

- **Idealized neuron models**
  - Idealization removes complicated details that are not essential for understanding the main principles.
  - It allows us to apply mathematics and to make analogies.
- **Break the limitations on MP Neuron**
  - What about non-boolean inputs (say, real number)? ✔
  - What if we want to assign more weight (importance) to some inputs? ✔
  - What about functions which are not linearly separable ? ✔
  - Do we always need to hand code the threshold? ✔

Week 3: From MLP to CNN

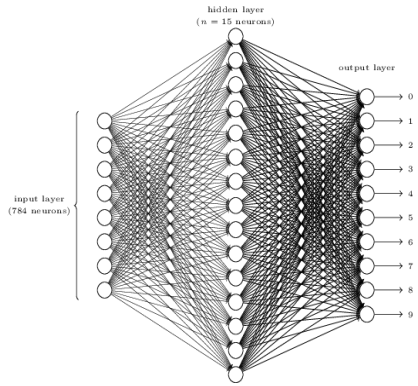


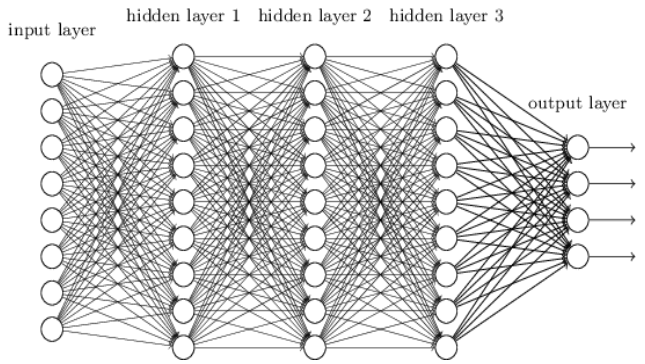
Agenda

- Shallow vs deep neural networks
- Vanishing gradient problem for deep networks
- Deep convolutional neural networks (CNN)
- Max-pooling layer
- A practical CNN example
- The impact of GPU on deep learning
- ImageNet and Large-Scale Visual Recognition Challenge
- Different deep neural nets (DNN)
- Summary of recent techniques for DNN
- Conclusions

Shallow vs deep neural networks

- Shallow neural network: ANN with only one hidden layer
- Deep neural network: ANN with more than one hidden layer





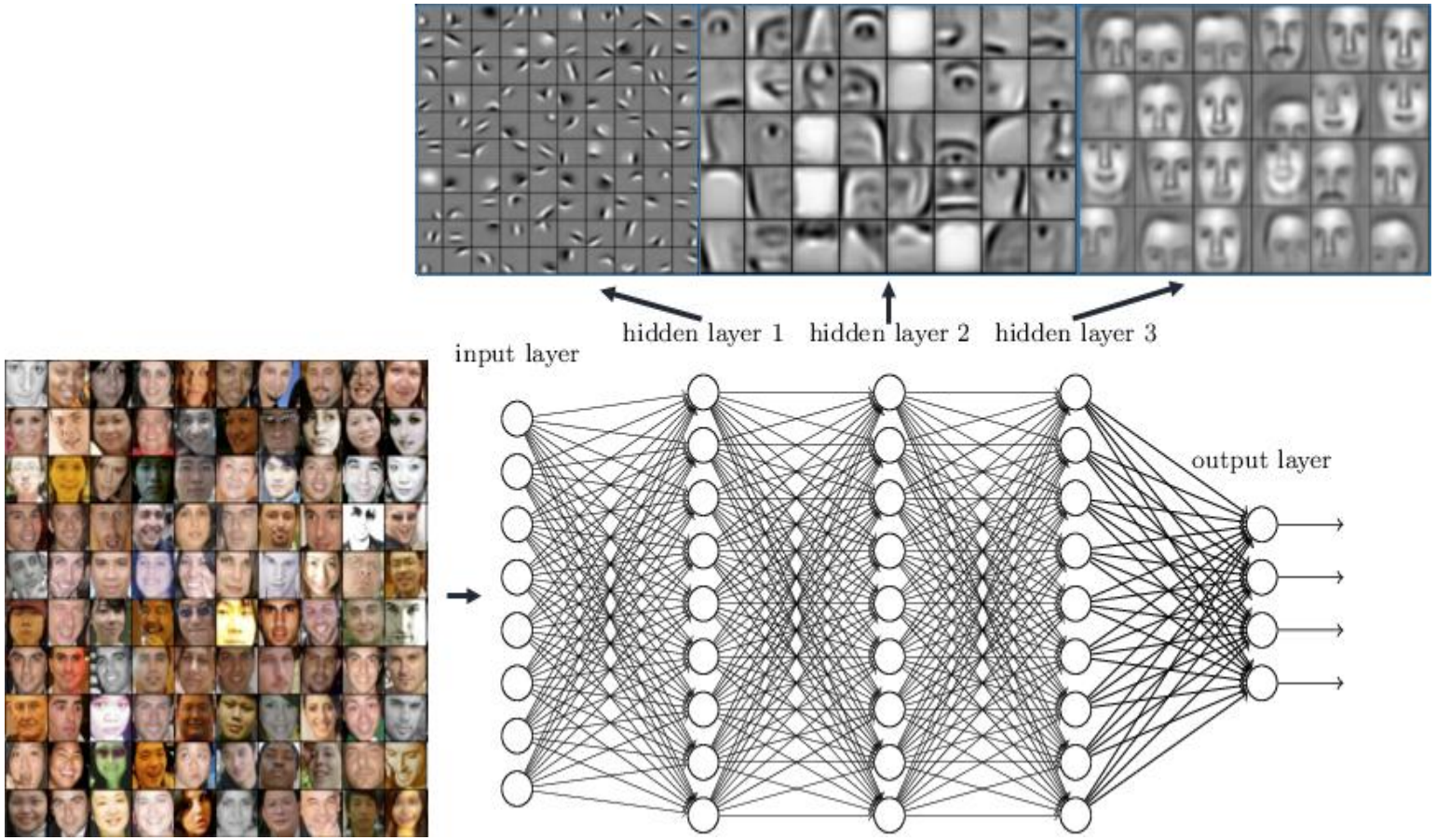
- There are some theoretical results suggesting that deep networks are intrinsically more powerful than shallow networks

  For certain problems and network architectures, this is proved in “On the number of response regions of deep feed forward networks with piece-wise linear activations”, by Razvan Pascanu, Guido Montúfar, and Yoshua Bengio (2014).

  See also the more informal discussion in section 2 of “Learning deep architectures for AI,” by Yoshua Bengio (2009)

Graphical illustration of the power of deep networks

- Deep neural networks learn hierarchical feature representations



[http://www.rsipvision.com/exploring-deep-learning/]

Can we train deep networks same as shallow networks?

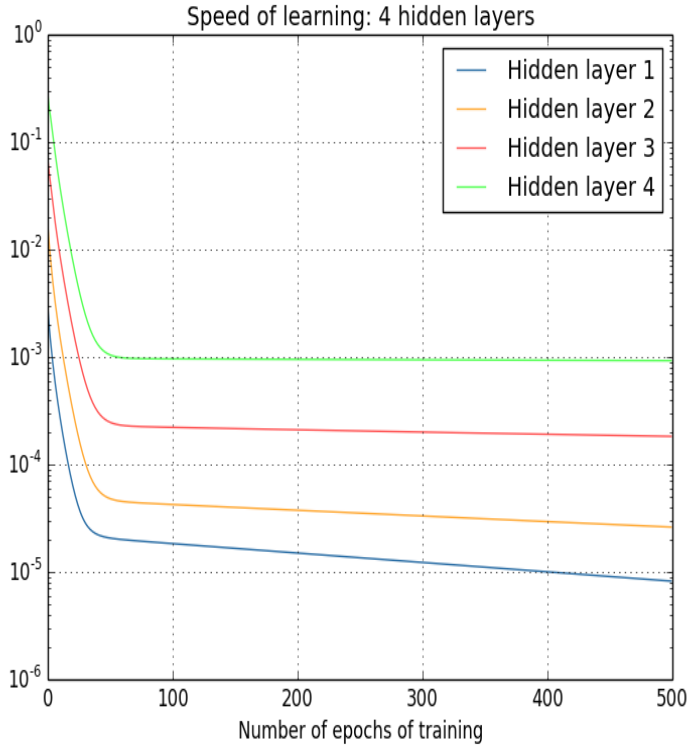
- Empirical results have shown that the deep networks not performing much (if at all) better than shallow networks
- That's why research in neural network has almost stopped for years before it became hot again recently

Agenda

- Shallow vs deep neural networks
- Vanishing gradient problem for deep networks
- Deep convolutional neural networks (CNN)
- Max-pooling layer
- A practical CNN example
- The impact of GPU on deep learning
- ImageNet and Large-Scale Visual Recognition Challenge
- Different deep neural nets (DNN)
- Summary of recent techniques for DNN
- Conclusions

Why is it hard to train deep networks?

- Different layers in our deep network are learning at vastly different speeds
  –When later layers in the network are learning well, early layers often get stuck during training, learning almost nothing at all



Use the 2-norm of gradients at each layer as a rough measures of the learning speed

Vanishing gradient problem for deep networks

- Vanishing gradient problem: the gradient tends to get smaller as we move backward through the hidden layers

  This means that neurons in the earlier layers learn much more slowly than neurons in later layers

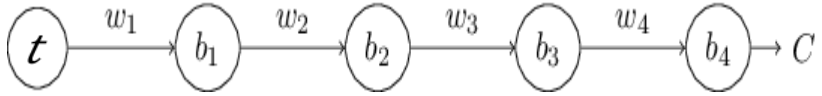
  The opposite phenomenon can also occur: the early layers may be learning well, but later layers can become stuck

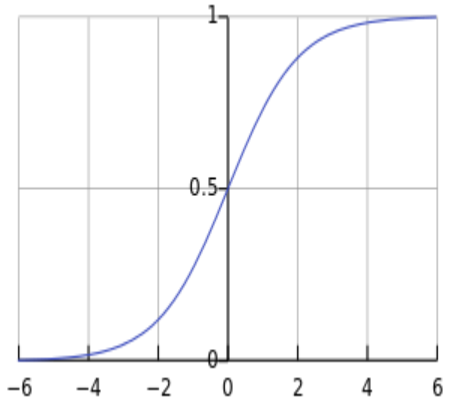
- This stuckness isn't simply due to bad luck, and there are fundamental reasons that the learning slowdown occurs
- There's an intrinsic instability associated to learning by gradient descent in deep, many-layer neural networks

Unstable gradients in deep neural nets

- Take a simple example with sigmoid neurons as an illustration

$$\frac{\partial C}{\partial b_1} = g'(z_1)w_2 g'(z_2)w_3 g'(z_3)w_4 g'(z_4) \cdot \frac{\partial C}{\partial o_4}$$



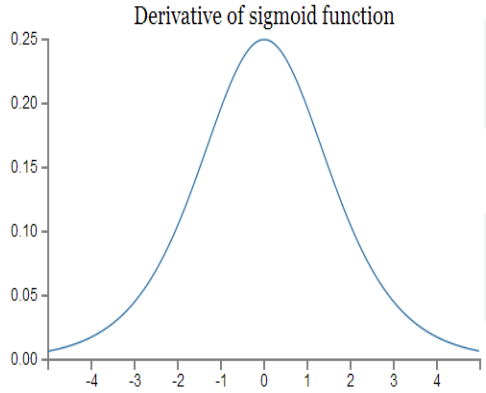


Logistic function: $g(z) = \frac{1}{1+e^{-z}}$

- The gradients (products) tend to exponentially decrease
  The more terms, the smaller the product

g'(z)≤0.25

$|w|<1$ most likely from initialization



Why is it hard to train deep networks?

- Besides vanishing gradients issues, there are many other obstacles
- It is still an active research area
- But the good news is that researchers have developed some techniques to overcome (to some extent) these obstacles

Agenda

- Shallow vs deep neural networks
- Vanishing gradient problem for deep networks
- Deep convolutional neural networks (CNN)
- Max-pooling layer
- A practical CNN example
- The impact of GPU on deep learning
- ImageNet and Large-Scale Visual Recognition Challenge
- Different deep neural nets (DNN)
- Summary of recent techniques for DNN
- Conclusions

Deep convolutional neural networks (CNN)

- One of the most widely used types of deep network
- Fully-connected nets treat far apart input pixels same as those close by
  - –Hence spatial information must be inferred from the training data
- In contrast, CNN proposes an architecture that inherently tries to take advantage of the spatial structure
  - –Such an architecture makes convolutional networks fast to train
  - –This, in turn, helps us train even deeper, many-layer networks
- Today, deep convolutional networks or some close variants are used in solving many interesting problems that go beyond image classification
- We will use image classification as a driving use case to explain the main concepts behind CNN

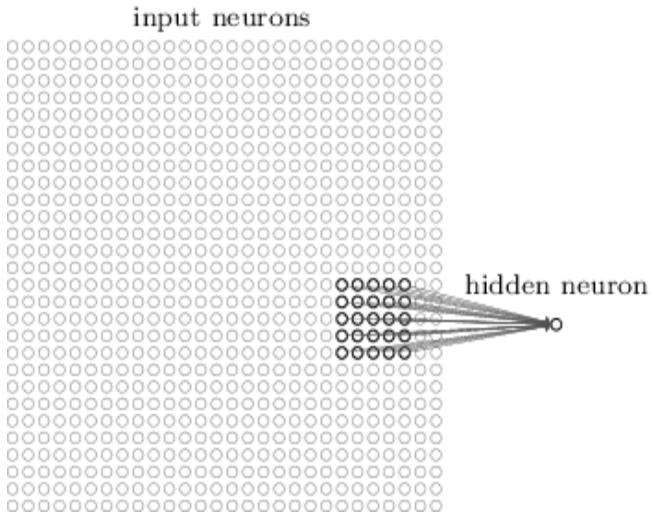
Three key ideas behind CNN

- Local receptive fields
- Shared weights

- Pooling (subsampling)

Local receptive fields

- Instead of connecting every input pixel to every hidden neuron, CNN only makes connections in small, localized regions of the input image



- That region is called the local receptive field for the hidden neuron

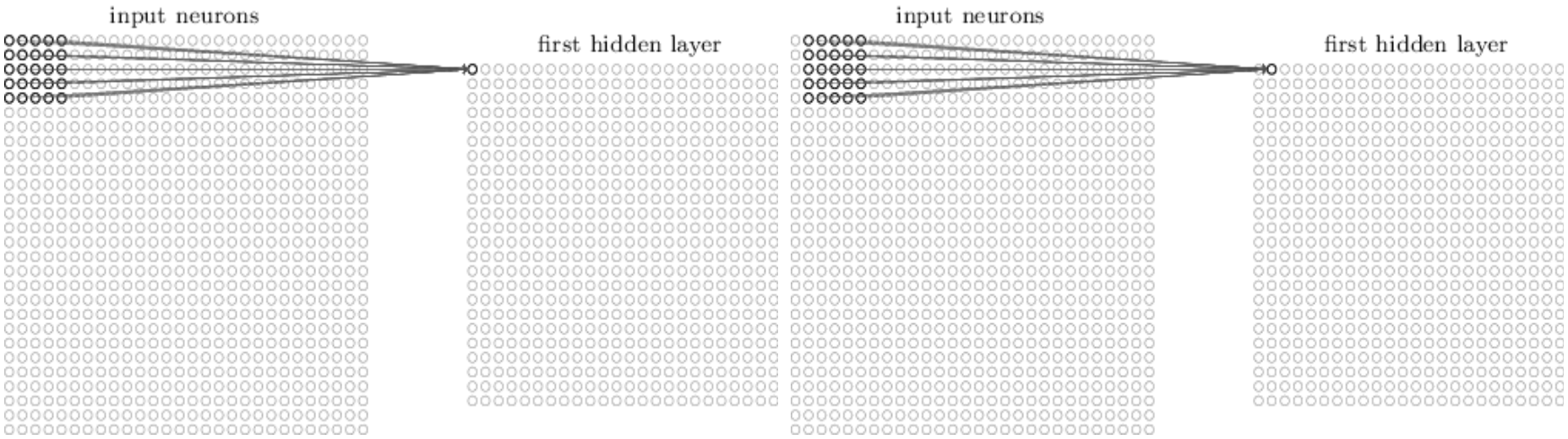
  –The corresponding hidden neuron learns the feature from its particular local receptive field

- The size and the number of local receptive fields are all hyperparameters

Sliding the local receptive field

- By sliding the local receptive field across the entire input image, we obtain different hidden neurons in the first hidden layer

  For example, sliding local receptive field one pixel at a time (stride length = 1)

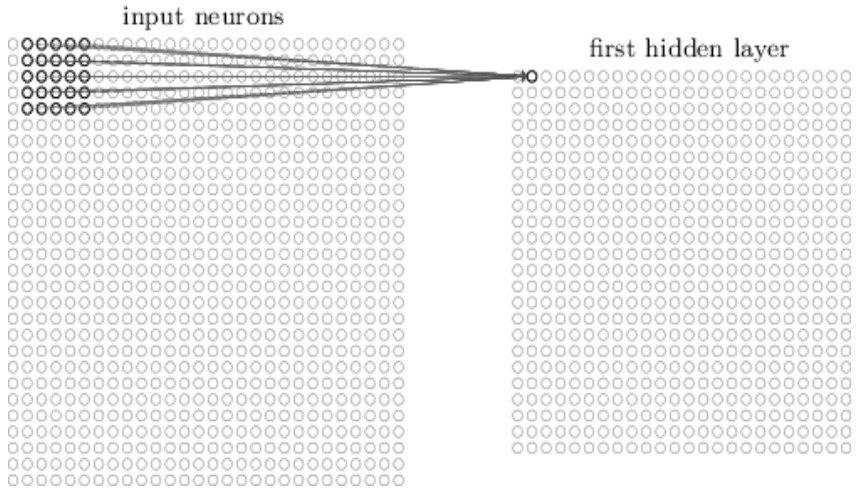


For example, if we have a 28×28 input image, and 5×5 local receptive fields, then there will be 24×24 neurons in the hidden layer

Stride length greater than 1 is also possible (another hyperparameter)

Shared weights and biases

- The weights and bias used to connect local fields to hidden neurons are the same for all hidden neurons



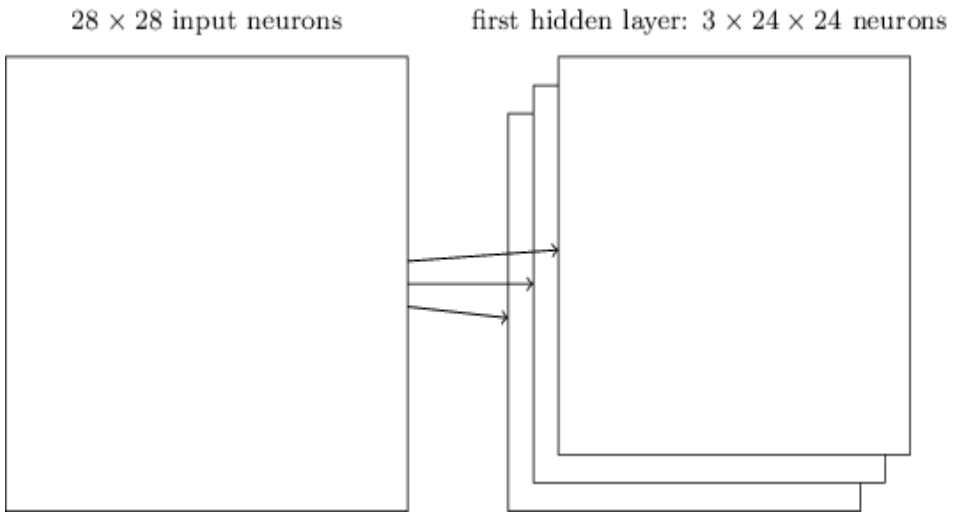
$$o_{j,k} = g(b + \sum_{l=0}^{4} \sum_{m=0}^{4} w_{l,m} t_{j+l,k+m})$$

The above operation is also called “convolution”

- All hidden neurons at the same layer detect exactly the same feature
  - –CNN are well adapted to the translation invariance of images
    - Move a picture of a cat (say) a little ways, and it's still an image of a cat
- Benefits of sharing weights and biases: greatly reduced number of parameters

Feature map

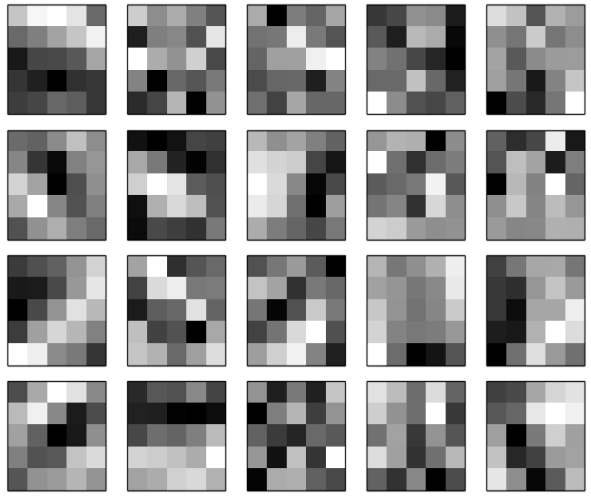
- We call the map from the input layer to the hidden layer a feature map
- The shared weights and bias are often said to define a kernel or filter
- A complete convolutional layer may consist of several different feature maps

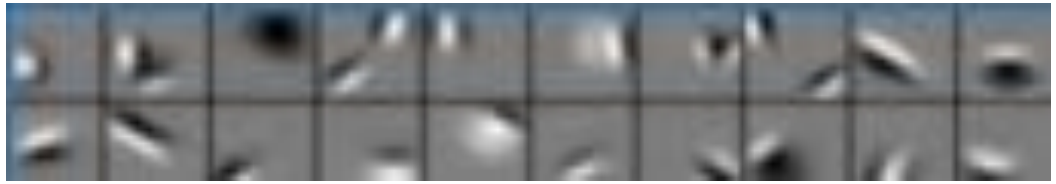


- Different feature maps learn different features from the inputs

What do the feature maps look like?

- The 20 images correspond to 20 different kernels
  - Each image corresponds to the 5×5 weights in the local receptive field
  - Whiter blocks mean a smaller (typically, more negative) weight, so the feature map responds less to corresponding input pixels
  - Darker blocks mean a larger weight, so the feature map responds more to the corresponding input pixels





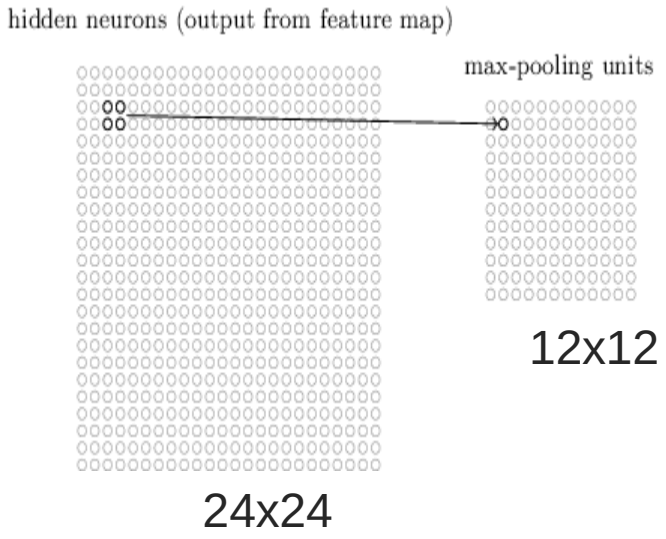
- It's clear there is a non-random spatial structure with clear sub-regions of light and dark (but it is not exactly know what they are)

Agenda

- Shallow vs deep neural networks
- Vanishing gradient problem for deep networks
- Deep convolutional neural networks (CNN)
- Max-pooling layer
- A practical CNN example
- The impact of GPU on deep learning
- ImageNet and Large-Scale Visual Recognition Challenge
- Different deep neural nets (DNN)
- Summary of recent techniques for DNN
- Conclusions

Pooling layers: summarize a region of neurons from its previous layer

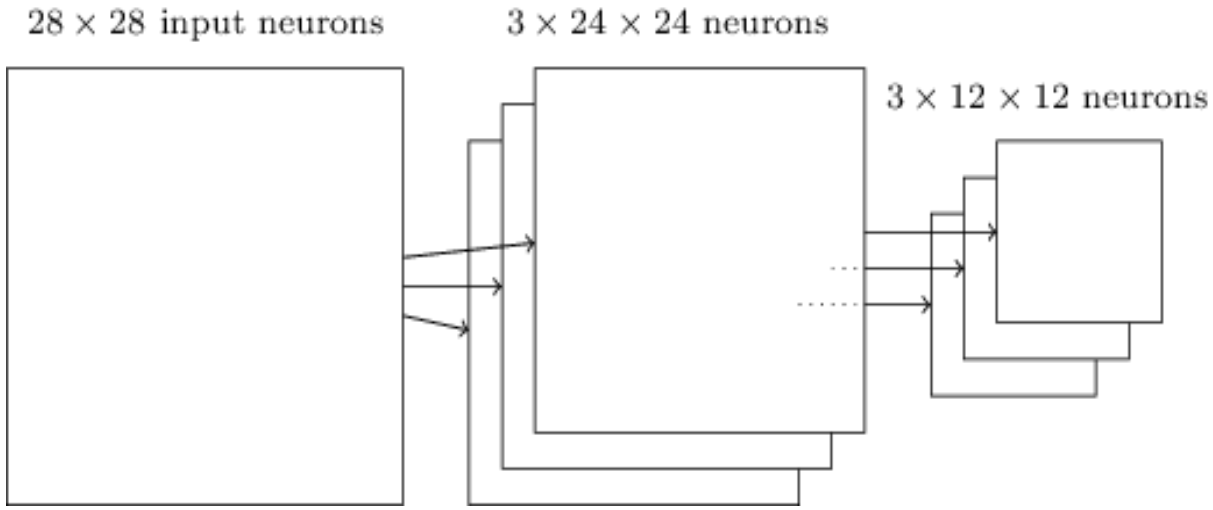
- One common pooling layer is the max-pooling layer
  - The pooling unit outputs the maximum value of the region's neurons
  - Regions are tiled from its previous layers
- In CNN, pooling typically used right after convolution layers
  - It takes each feature map output from the convolutional layer and prepares a condensed feature map
    - E.g., if the region is defined as 2×2, the max-pooling unit will output the maximum value in the 2x2 input region



Max-pooling and convolutional layers

- The convolutional layer usually involves more than a single feature map, so we apply max-pooling to each feature map separately

  For example, if there were three feature maps, the combined convolutional and max-pooling layers would look like



Intuition behind pooling layers

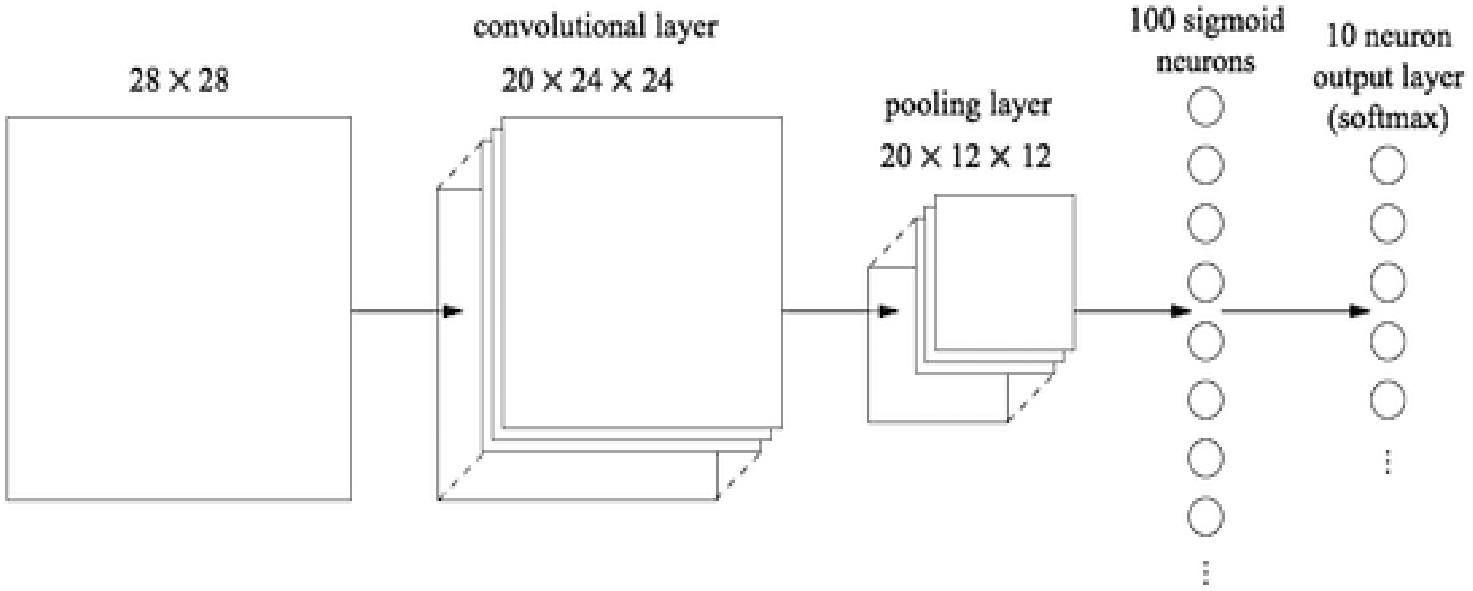
- Max-pooling finds a particular feature anywhere in a region of the image, and then throws away the exact positional information
- The intuition is that once a feature has been found, its exact location isn't as important as its rough location relative to other features
- The benefit is much reduced features after pooling, thus reducing the number of parameters needed in later layers
- Other pooling possible, such as L2-pooling, which takes the square root of the sum of the squares of the region’s neurons

Agenda

- Shallow vs deep neural networks
- Vanishing gradient problem for deep networks
- Deep convolutional neural networks (CNN)
- Max-pooling layer
- A practical CNN example
- The impact of GPU on deep learning
- ImageNet and Large-Scale Visual Recognition Challenge
- Different deep neural nets (DNN)
- Summary of recent techniques for DNN
- Conclusions

Putting things together for a practical CNN

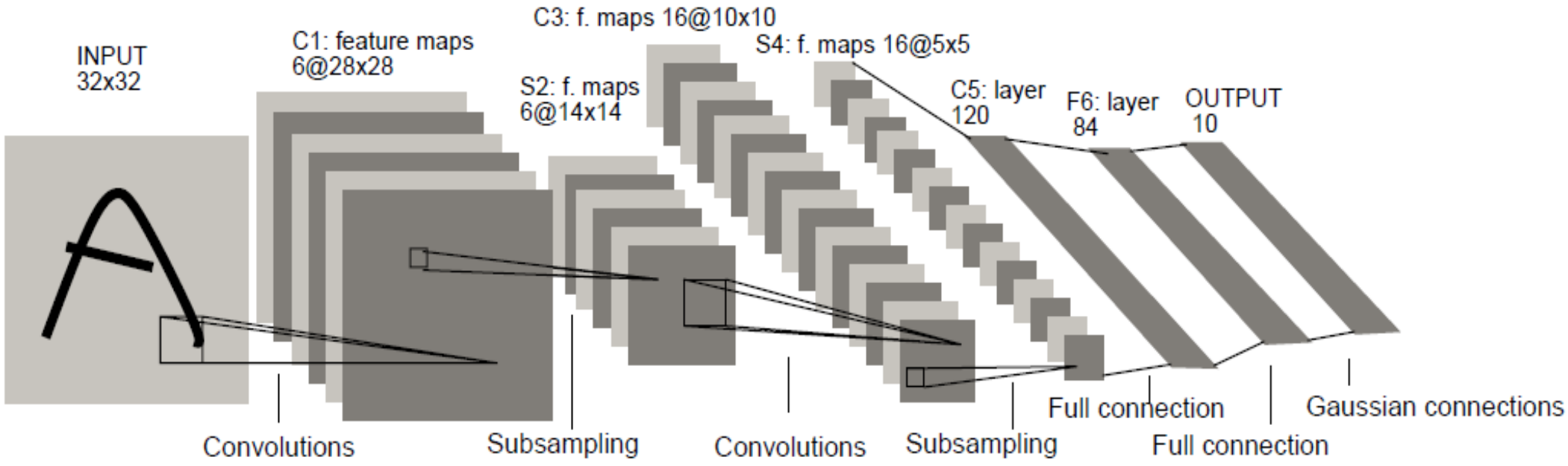
- An example for the MNIST digits classification



- A practical CNN may use many more convolutional layers

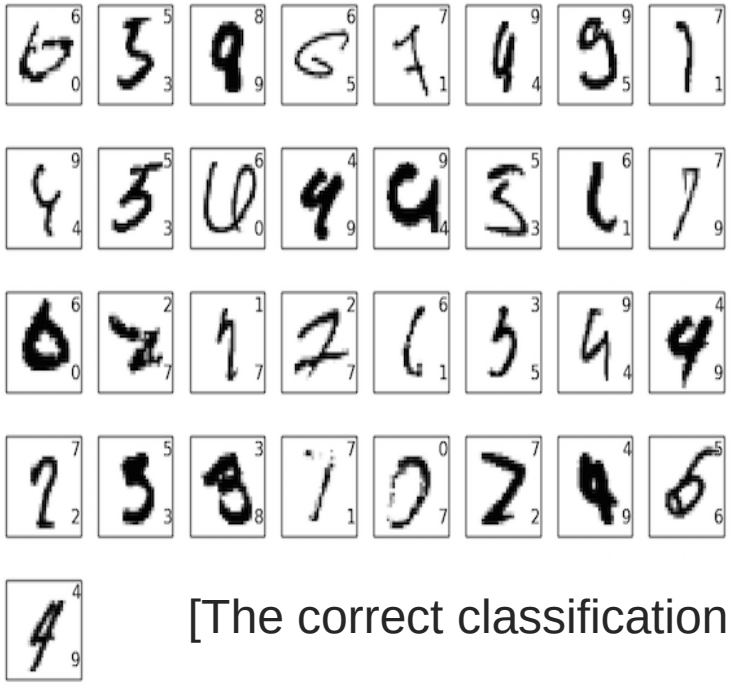
LeNet

- The most known CNN for recognizing handwritten digits [LeCun et al., 1998]



A more fine tuned CNN accuracy for recognizing handwriting digits

- Test accuracy of 99.67%, i.e., 9,967 digits were correctly recognized out of 10,000 MNIST test images



[The correct classification is in the top right]

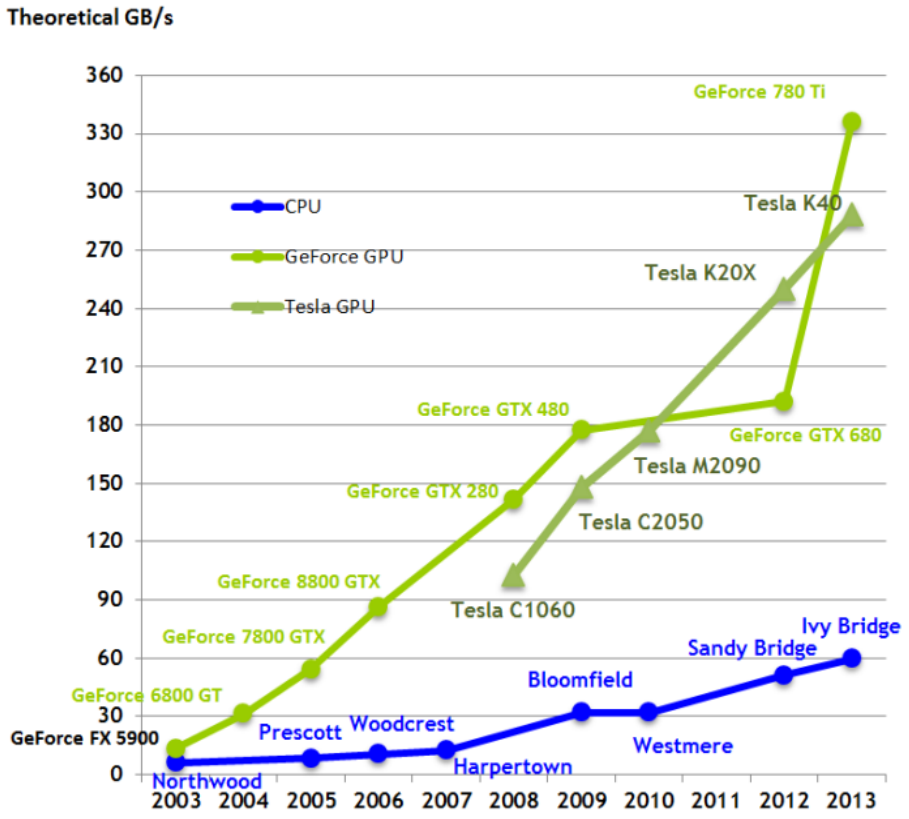
- State-of-the-art achieved accuracy of 99.79%
  - –See the following link for a list of state-of-the-art results
    - [http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html]

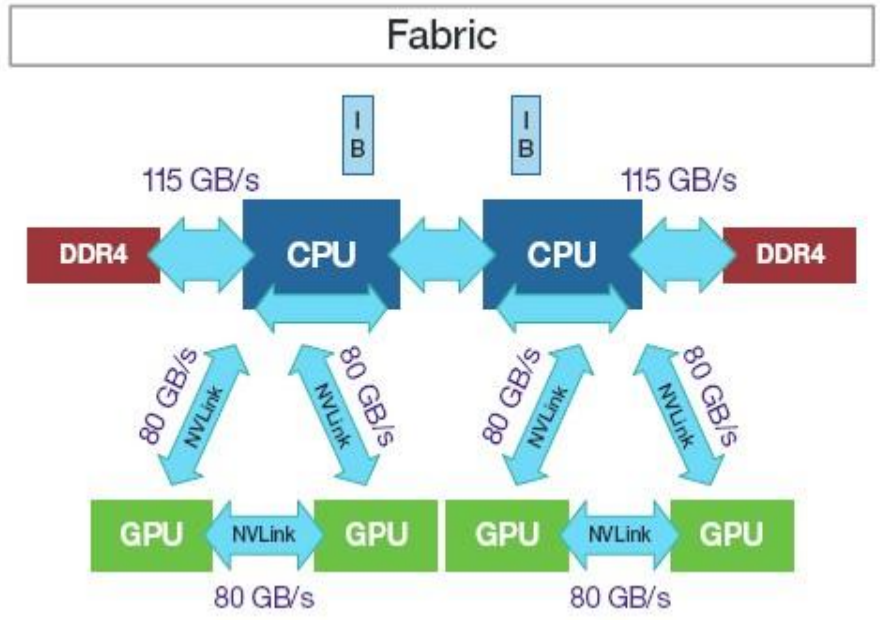
Agenda

- Shallow vs deep neural networks
- Vanishing gradient problem for deep networks
- Deep convolutional neural networks (CNN)
- Max-pooling layer
- A practical CNN example
- The impact of GPU on deep learning
- ImageNet and Large-Scale Visual Recognition Challenge
- Different deep neural nets (DNN)
- Summary of recent techniques for DNN
- Conclusions

Training DNN requires significant computation time

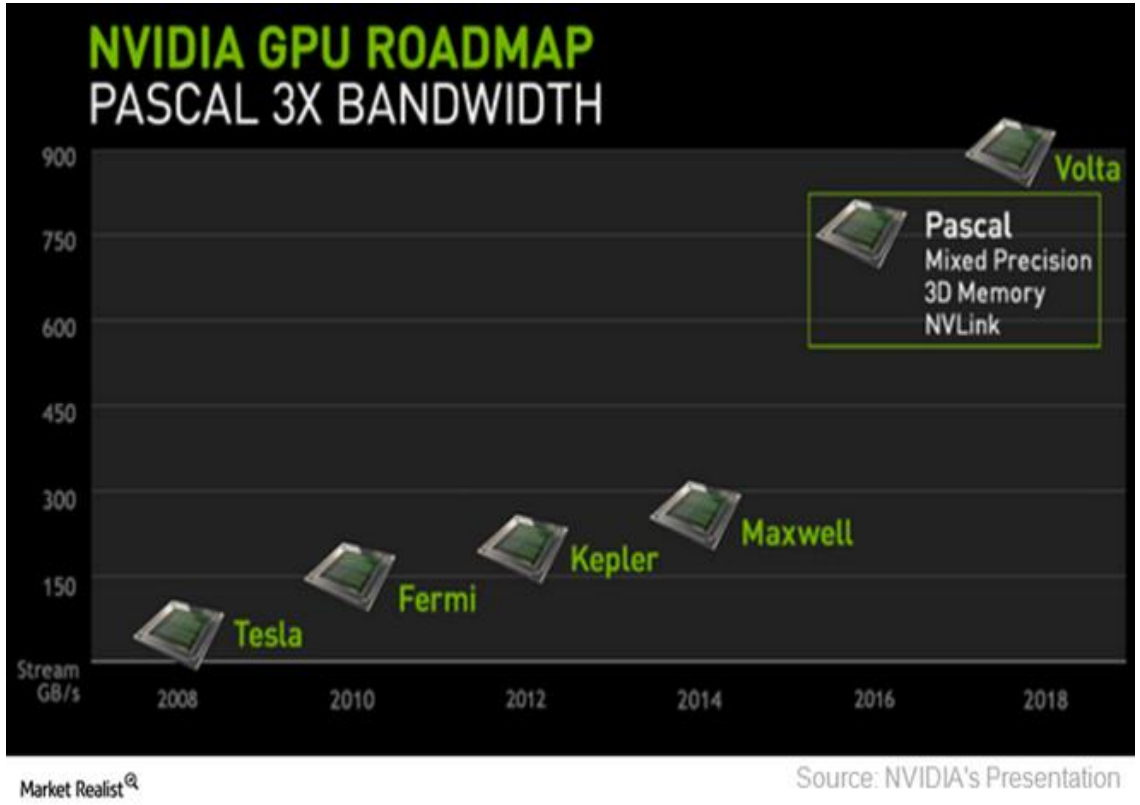
- When MNIST was first introduced in 1998, it took weeks to train a CNN on a then-state-of-the-art workstation
- Adoption of GPU for deep learning has changed this field forever





The impact of GPU on deep learning

- GPU provides the computation workhorse for training DNN



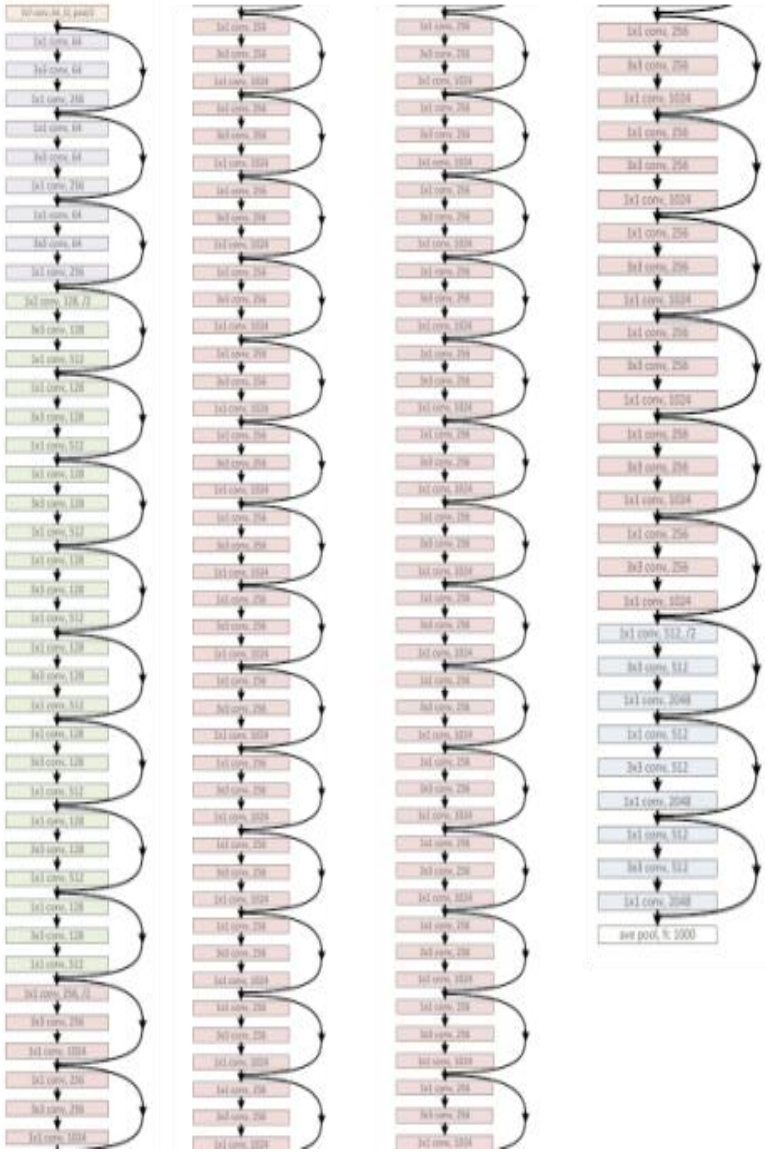
- Powerful GPU libraries made using GPU for DNN development easier –cuBlas, cuDNN

How deep the network should be?

- The number of layers in a DNN just gets bigger and bigger

  –State-of-the-art easily goes beyond 150 layers (even 1000+ for natural language processing)

- The turning point for DNN was to find practical ways to training networks that go beyond the shallow 1- and 2-hidden layer networks around the mid-2000s



He, K., Zhang, X., Ren, S., & Sun, J., Deep Residual Learning for Image Recognition (2015)

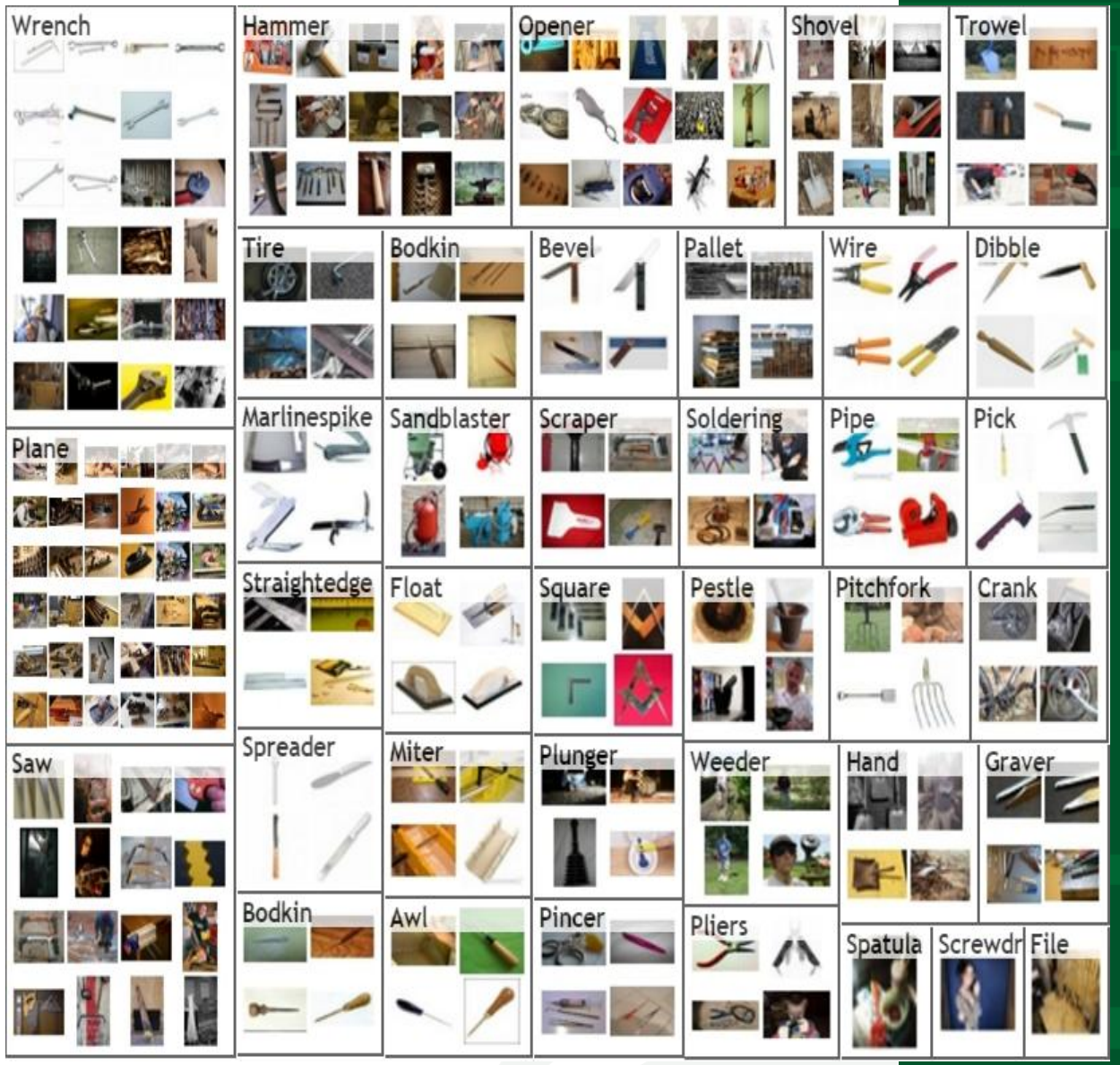
Agenda

- Shallow vs deep neural networks
- Vanishing gradient problem for deep networks
- Deep convolutional neural networks (CNN)
- Max-pooling layer
- A practical CNN example
- The impact of GPU on deep learning
- ImageNet and Large-Scale Visual Recognition Challenge
- Different deep neural nets (DNN)
- Summary of recent techniques for DNN
- Conclusions

ImageNet

- ImageNet was created in 2009 by collecting millions of images from the open net, and labeled the images using crowd sourcing method (via Amazon's Mechanical Turk service)

- The 2011 ImageNet data contains 16 million full color images, in 20 thousand categories

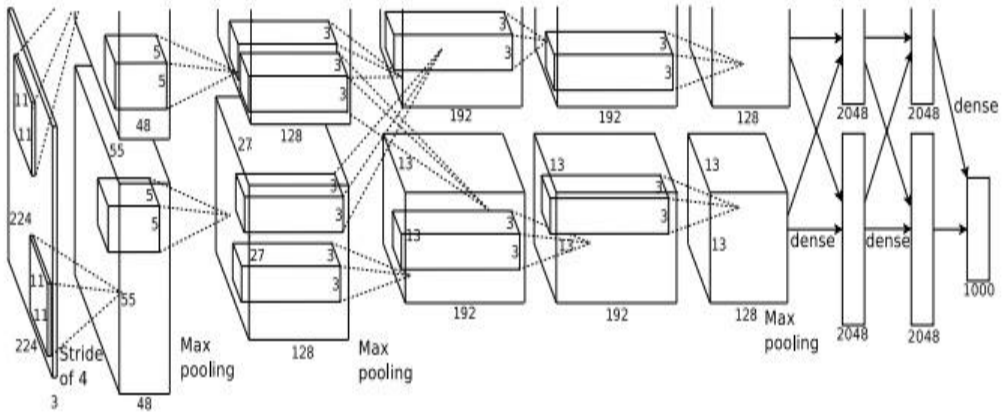


ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)

- Since 2010, the annual ILSVRC competition attracted research teams to submit programs to classify and detect objects and scenes
- The breakthrough came from a paper from ILSVRC-2012

  –Where training set contained about 1.2 million ImageNet images, drawn from 1,000 categories

  –The validation and test sets contained 50,000 and 150,000 images, respectively, drawn from the same 1,000 categories

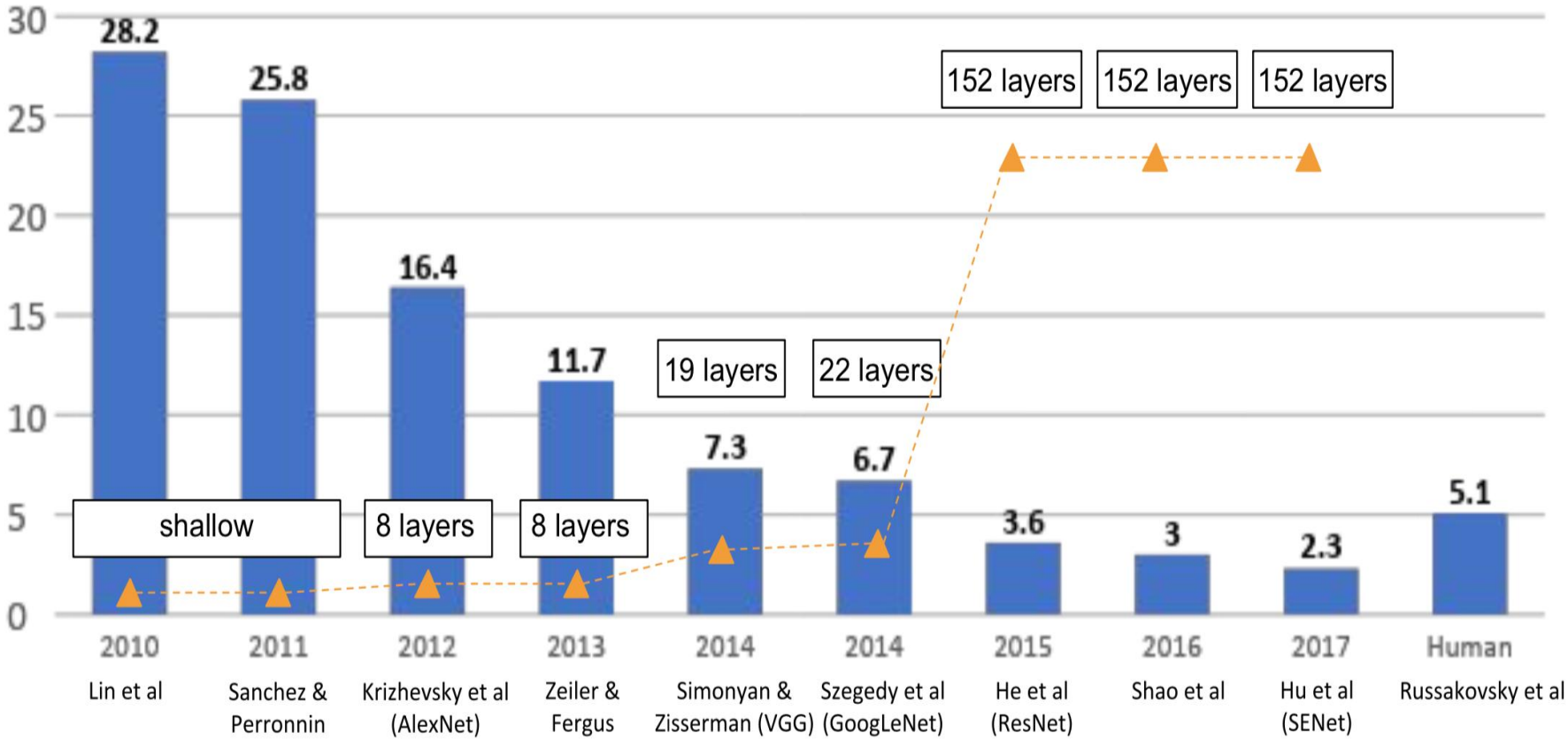


[ImageNet classification with deep convolutional neural networks, by Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton (2012).]

Agenda

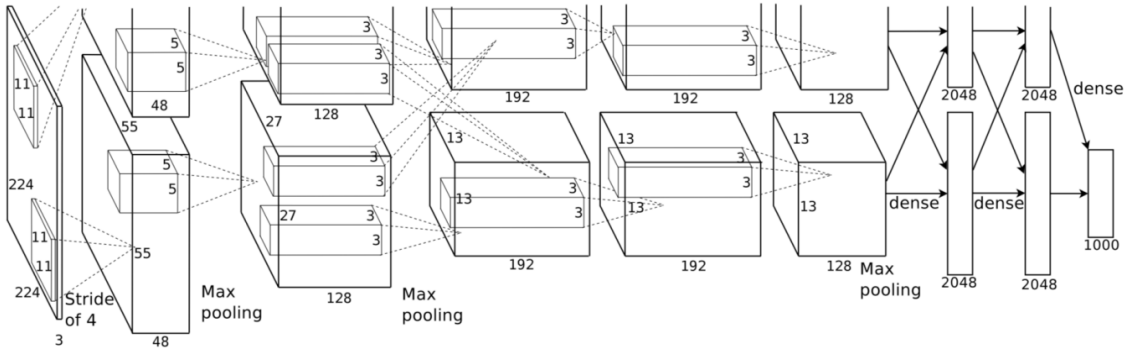
- Shallow vs deep neural networks
- Vanishing gradient problem for deep networks
- Deep convolutional neural networks (CNN)
- Max-pooling layer
- A practical CNN example
- The impact of GPU on deep learning
- ImageNet and Large-Scale Visual Recognition Challenge
- Different deep neural nets (DNN)
- Summary of recent techniques for DNN
- Conclusions

ILSVRC Winners



AlexNet

- AlexNet is the first CNN model which achieves great success in image classification tasks.
- It is spread over 2 GPUs with each one containing half of the channels.



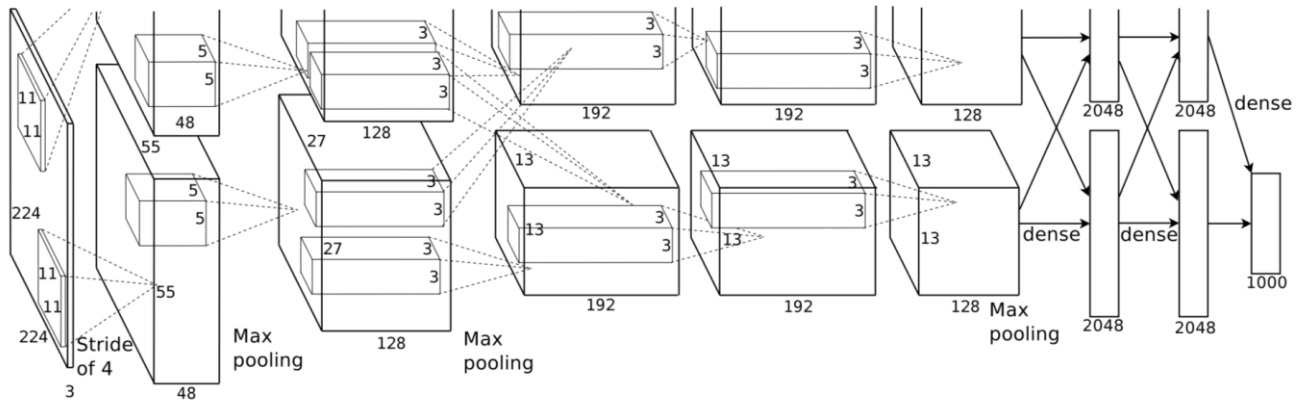
Features:

- Use larger kernels (11x11) for feature extraction.
- The first architecture to use ReLU as non-linear activation.
- Use local response normalization (LRN) to speedup training.
- ImageNet top-5 error: 16.4%

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." (2012)

AlexNet

- AlexNet is the first CNN model which achieves great success in image classification tasks.
- It is spread over 2 GPUs with each one containing half of the channels.



How many computations (Multiply-Accumulates) are there in the first convolutional layer?

$11 \times 11 \times 3 \times 55 \times 55 \times 96 = 105.4M$

Features:

- Use larger kernels (11x11) for feature extraction.
- The first architecture to use ReLU as non-linear activation.
- Use local response normalization (LRN) to speedup training.
- ImageNet top-5 error: 16.4%

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." (2012)

Local Response Normalization (LRN)

- AlexNet used ReLU as opposed to the more common tanh and sigmoid at that time → unbounded output
- Encourage lateral inhibition
  - A concept in Neurobiology that refers to the capacity of a neuron to reduce the activity of its neighbors.
  - In DNNs, the purpose is to carry out local contrast enhancement so that locally maximum pixels values are used as excitation for the next layers

Local Response Normalization (LRN)

- AlexNet places LRN after the ReLU non-linearity in some layers.

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

- LRN normalizes the activations to speed up the training of neural networks.
- In practice, AlexNet determines the hyperparameters k, n, α, β by running experiments on a validation set. AlexNet uses $k = 2, n = 5, \alpha = 10^{-4}$, and $\beta = 0.75$.

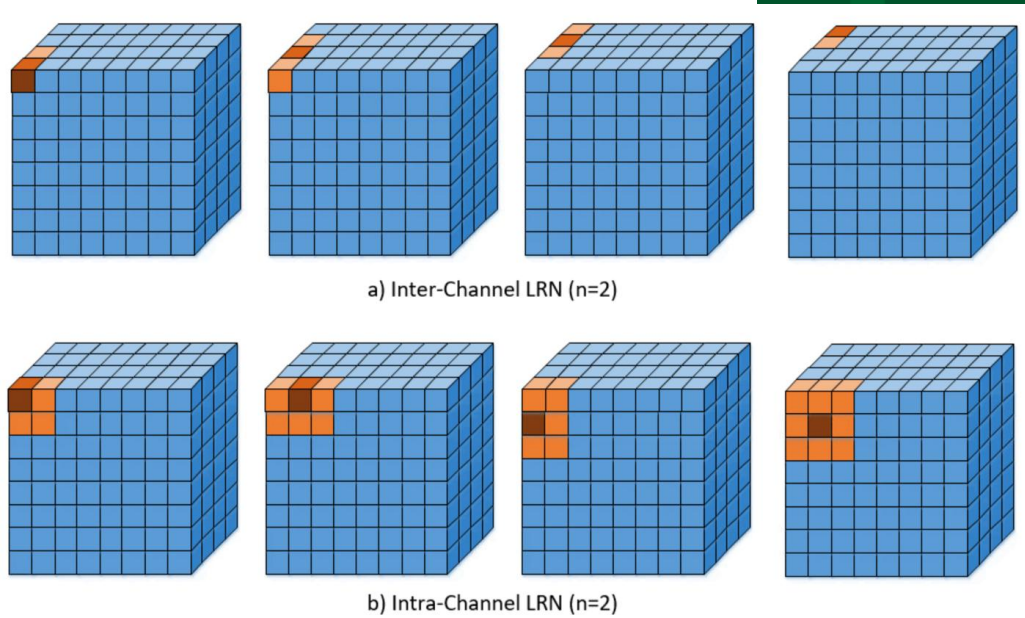
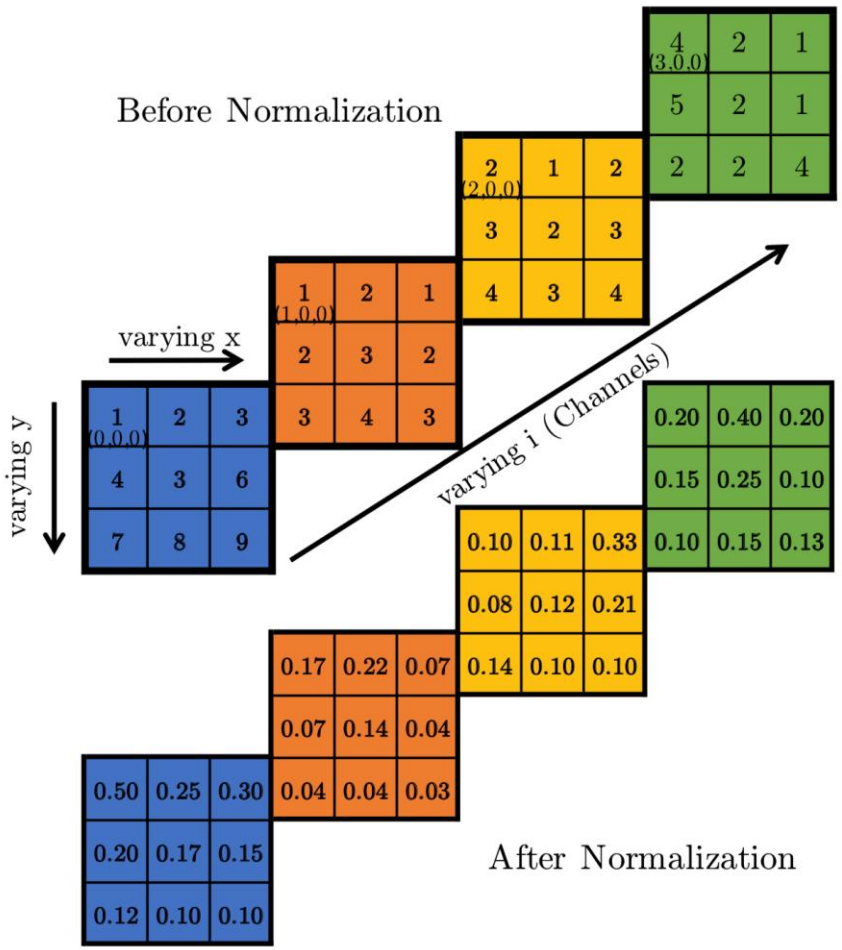


Illustration of Inter-Channel LRN



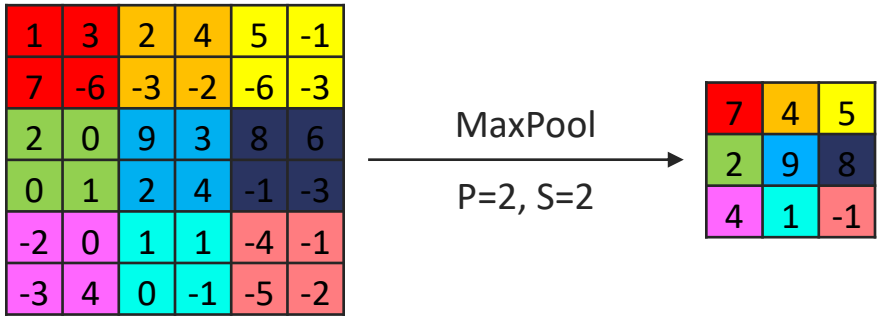
$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

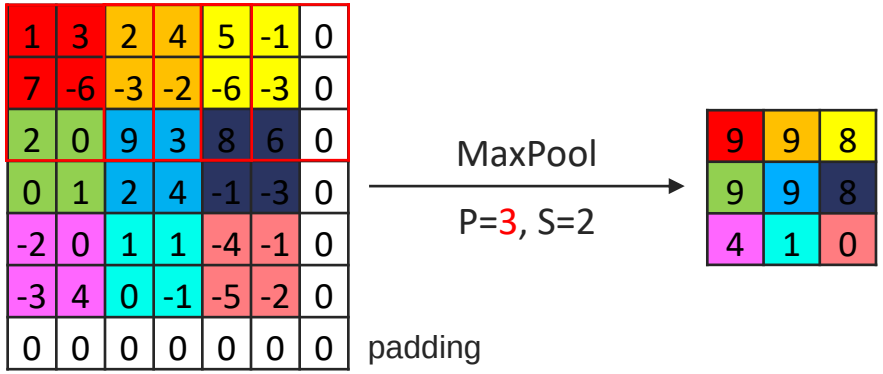
(k,α, β, n)=(0,1,1,2)

AlexNet

Overlapping pooling

- AlexNet sets a larger pool size in each pooling layer to conduct overlapping pooling. Overlapping pooling makes the model more difficult to overfit as it considers the significance of values over a joint pooling region.





Larger activation will dominate the down-sampled feature map.

AlexNet

- **How to train AlexNet?**
  - Batch size 128, initial learning rate 0.01, weight decay 5e-4
  - Use SGD Momentum with momentum 0.9
  - Use normalization layers (LRN)
  - Dropout 0.5 for FC layers (except final FC).
  - Use aggressive data augmentation (Shifting, flipping, etc.).
- Later training approaches inherits most of the AlexNet approach.

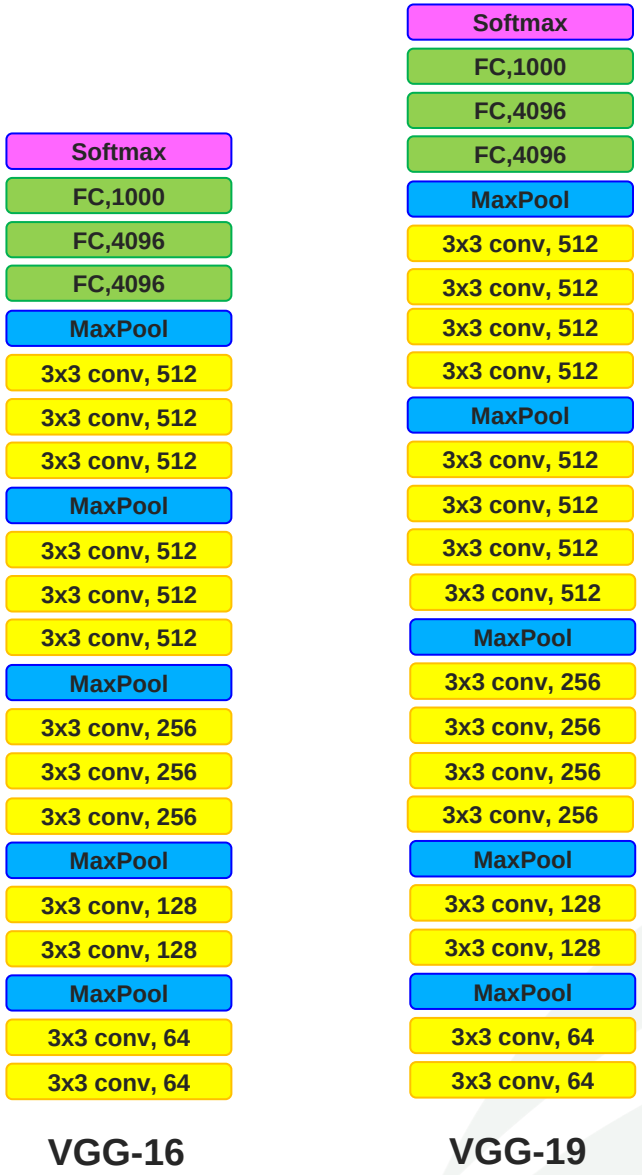
VGG

- VGG is the first architecture to create very deep convolutional neural networks. (11-19 layers)

Features:

- Efficient 3x3 convolution.
- ImageNet top-5 error: 7.3%
- Until now, VGG model is still good enough for transfer learning into other tasks (e.g., object detection).
- Very large model.

VGG 16/19 has 140/144 Million parameters and 16G/20G Multiply-Accumulates (MACs) in total.

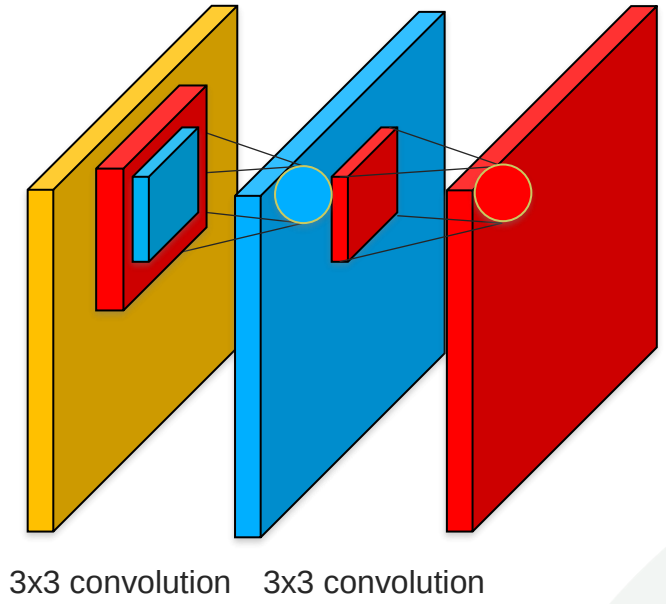


Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." (2014)

VGG

Recap: Receptive field

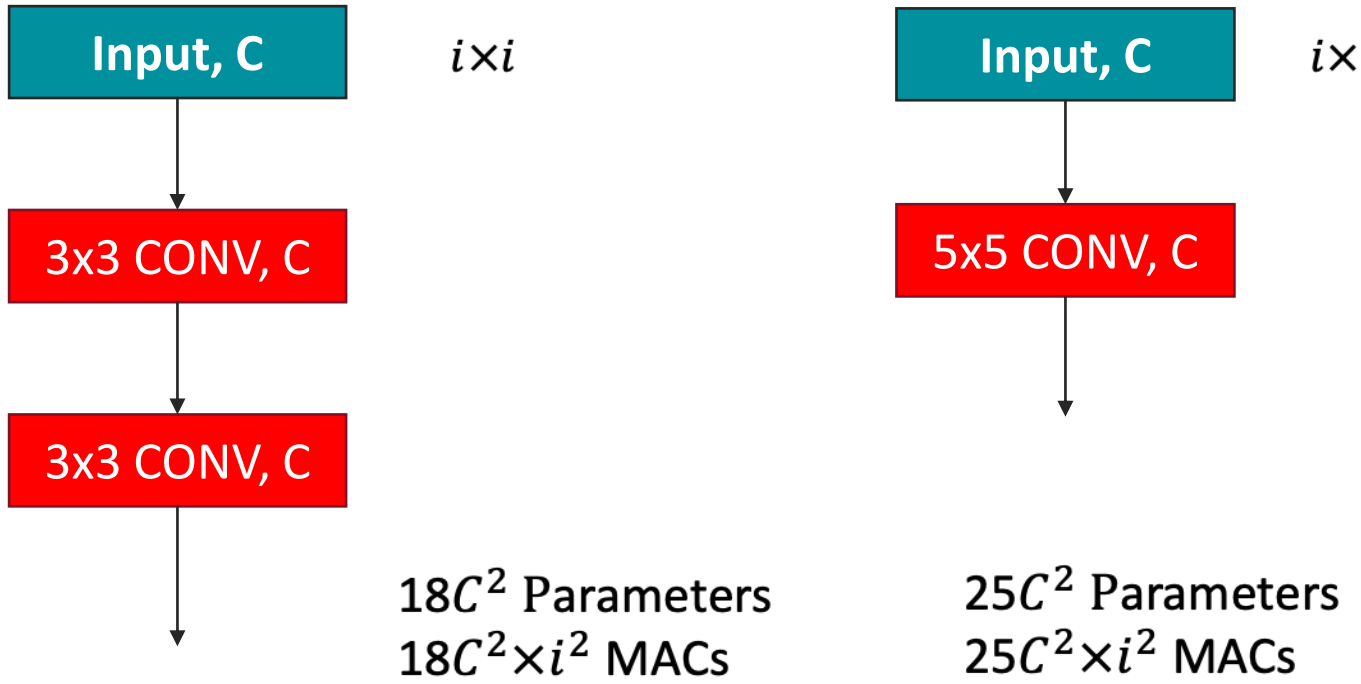
- The output of the red neuron is affected by the red region in the first orange input feature map. Thus, the **receptive field** for these two 3x3 convolution is of dimension 5x5.



VGG

- VGG only uses 3x3 convolution for convolutional layers.

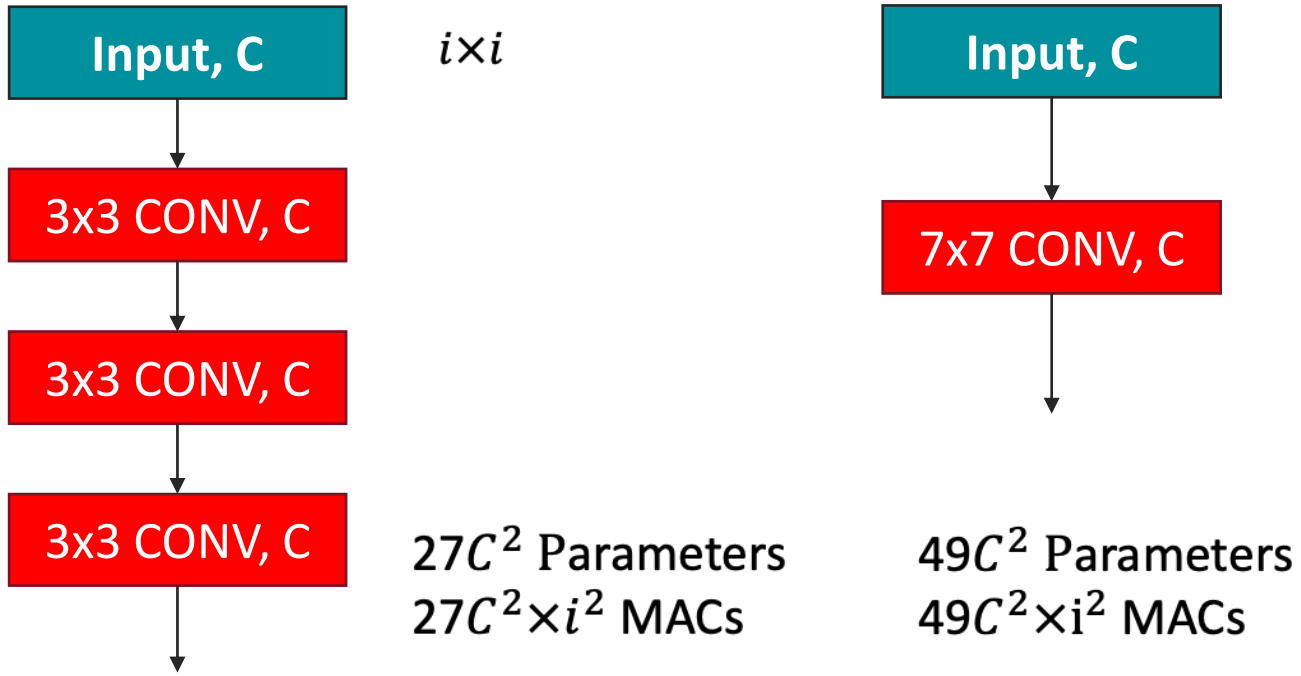
  2 3x3 convolution has the same receptive field as 1 5x5 convolution with fewer parameters and MACs.



VGG

- VGG only uses 3x3 convolution for convolutional layers.

  3 3x3 convolution has the same receptive field as 1 7x7 convolution with fewer parameters and MACs.



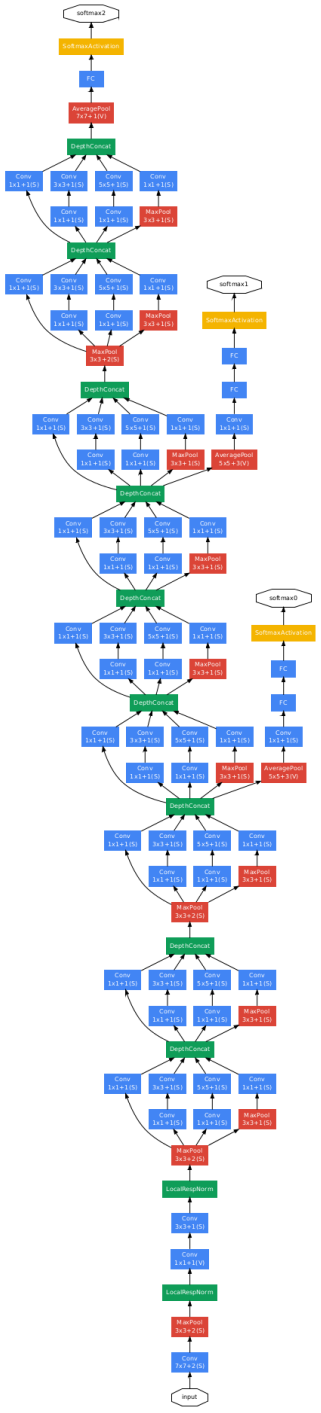
VGG

- How to train VGG?
  - Batch size 256, initial learning rate is 0.01, weight decay 5e-4.
  - Decay the learning rate by a factor of 0.1 if validation accuracy plateaus.
  - Use SGD Momentum with momentum 0.9.
  - Dropout 0.5 for FC layers (except final FC).
  - Data preprocessing is a bit different from AlexNet. See the VGG paper for more details.
  - Use model ensembling to reach better results.

Designing CNNs in a nutshell.

Fun fact, this meme was referenced in the first inception net paper.

https://knowyourmeme.com/memes/we-need-to-go-deeper



Designing CNNs in a nutshell.

Fun fact, this meme was referenced in the first inception net paper.

https://knowyourmeme.com/memes/we-need-to-go-deeper

GoogLeNet (Inception)

- From GoogLeNet, convolutional neural networks not or evolve into multiple branches.

Features:

Build a deeper model with a mixture of convolution filte
Good at extracting multi-dimensional information.

Uses local response normalization (LRN) to speed up t

ImageNet top-5 error: 6.7%

Use bottleneck structure to reduce parameters.

Time and memory consuming. Take about a week to re

Szegedy, Christian, et al. “Going deeper with convolutions.’ (2015)

GoogLeNet (Inception)

- From GoogLeNet, convolutional neural networks not only evolve into multiple branches.

Features:

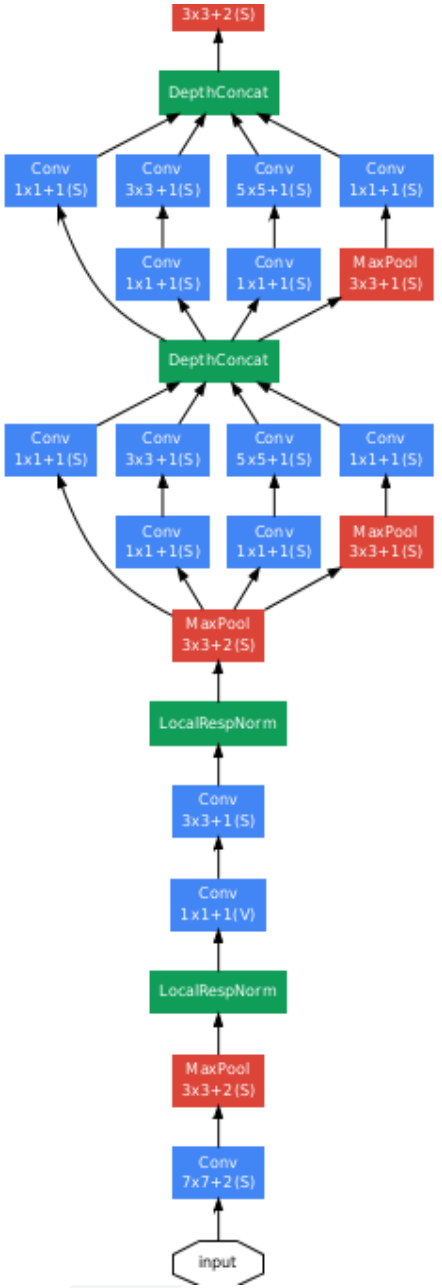
Build a deeper model with a mixture of convolution filters
Good at extracting multi-dimensional information.

Uses local response normalization (LRN) to speed up trai

ImageNet top-5 error: 6.7%

Use bottleneck structure to reduce parameters.

Time and memory consuming. Take about a week to reac

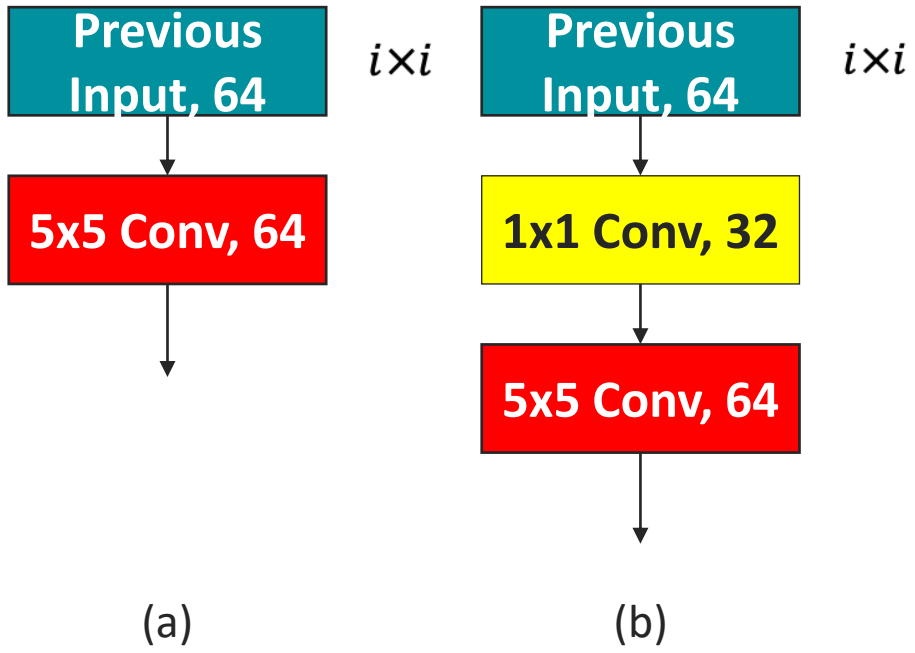


Szegedy, Christian, et al. “Going deeper with convolutions.’ (2015)

GoogLeNet

Bottleneck structure

- The number of filters is reduced before convolve large filters (e.g. 5x5). This structure is also called bottleneck structure.

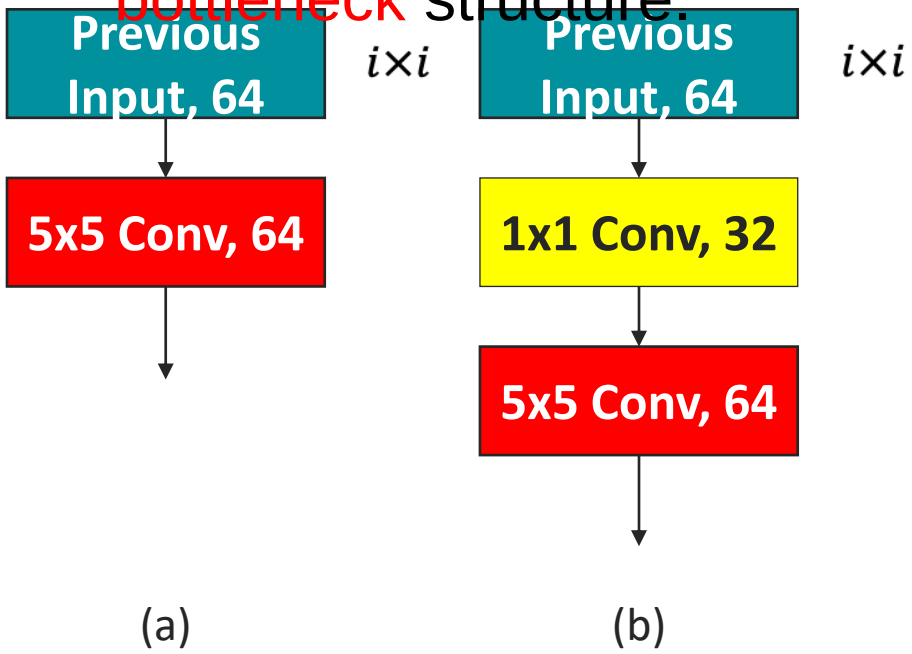


Q: For (a) and (b), calculate the weight parameter number and multiplications for each model.

GoogLeNet

Bottleneck structure

- The number of filters is reduced before convolve large filters (e.g. 5x5). This structure is also called bottleneck structure.



Q: For (a) and (b), calculate the weight parameter number and multiplications for each model.

A:

(a) $5\times5\times64\times64 + 64 =$ 102464 weight parameters

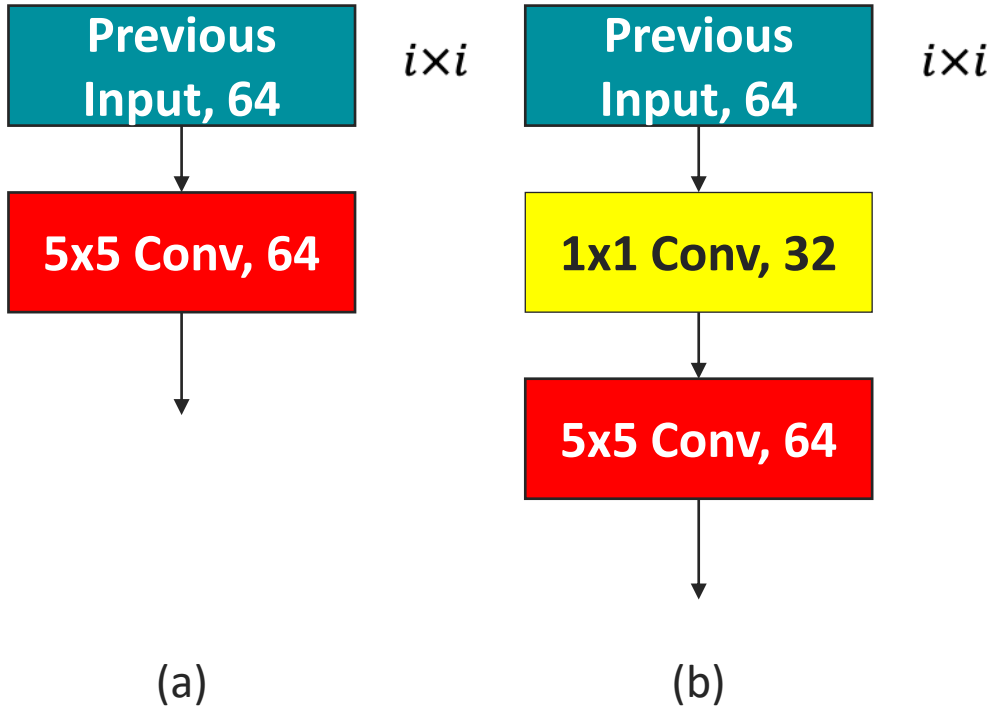
$5\times5\times64\times64\times i\times i = 102400\times i^2$ MACs

(b) $1\times1\times64\times32 + 32 +$ $5\times5\times32\times64 + 64 = 53344$ weight parameters

$(1\times1\times64\times32 + 5\times5\times32\times64)\times i^2 = 53248i^2$ MACs

GoogLeNet

- Bottleneck structure loses information by cutting down the number of feature maps. Does this information loss lead to performance degradation of GoogLeNet?

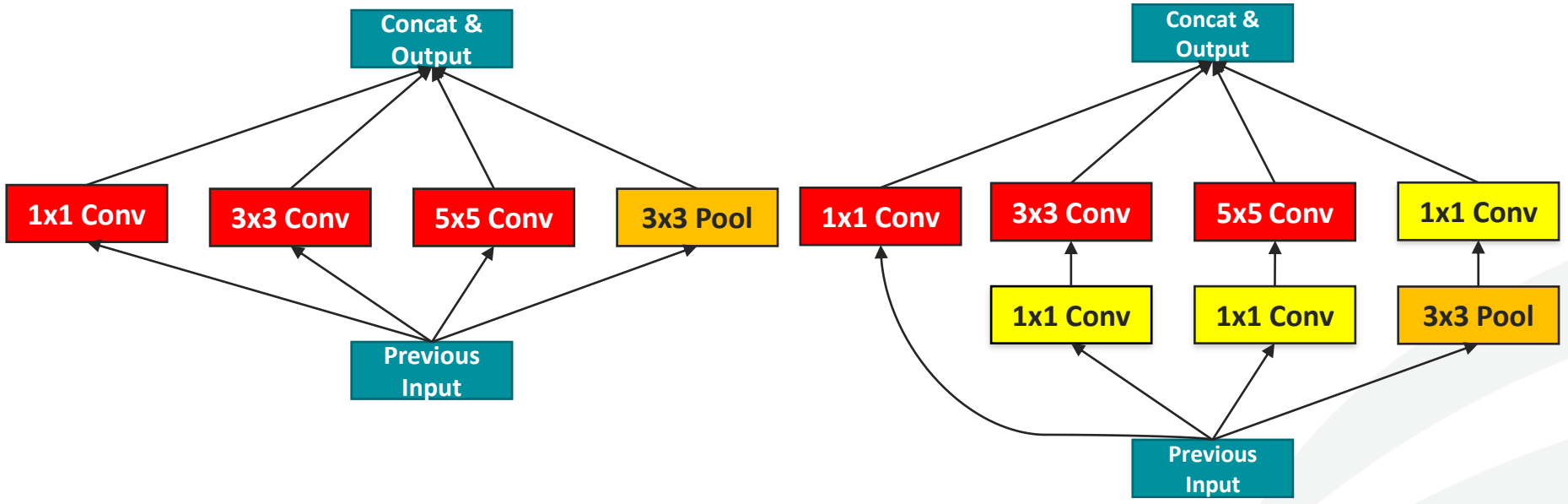


- Total information is not always proportional to the width of neural networks.
- In addition, project high-dimensional inputs into lower-dimensional space is also able to preserve information.
- This ‘projection’ may even act as a form of regularization to prevent overfitting, which improves the performance of GoogLeNet.

GoogLeNet

Bottleneck structure

- Here is a visualization of the bottleneck structure in one of the blocks in GoogLeNet.



Naïve GoogLeNet **GoogLeNet with dimension reduction**

GoogLeNet

- **How to train GoogLeNet?**

  Batch size is 256, initial learning rate is 0.01, decay the learning rate by a factor of 0.96 every 8 epochs.

  40% dropout before the final FC layer.

  Use SGD Momentum with momentum 0.9.

  Preprocessing is a bit different from VGG. See the GoogLeNet-V1 (V2) paper for details.

  Average predictions over multiple crops of the same input image.

Internal Covariate Shift (ICF)

- ICF arises due to the changing distribution of the hidden neurons/activation.

- We can do random sampling for batches; but still internal/hidden neurons have similar issues
- It can slow down the training
  –For a given batch a network tries to learn a certain distribution, which is different for the next batch.
  –Will keep on bouncing back and forth between distributions until it converges.

Batch Normalization (BN)

- A trainable layer to address ICF
- The normalization is carried out for each pixel across all the activations in a batch.

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

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

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

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

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

- Scaling and shifting is to let the training decide whether we even need the normalization or not.

Comparison between LRN and BN

- Both are called “normalization”, but are different
- LRN has multiple directions to perform normalization across (Inter or Intra Channel)
- BN has only one way of being carried out (for each pixel position across all the activations).

Comparison of the two normalization techniques in DNNs				
Norm Type	**Trainable**	**Training Parameters**	**Fouses on**	Regularization
LRN	No	0	Lateral Inhibition	No
BN	Yes	2 (Scale and Shift)	Internal Covariate Shift	Yes

GoogLeNet-V2 (Inception-V2)

Adding batch normalization operation before each nonlinearity in the Inception architecture.

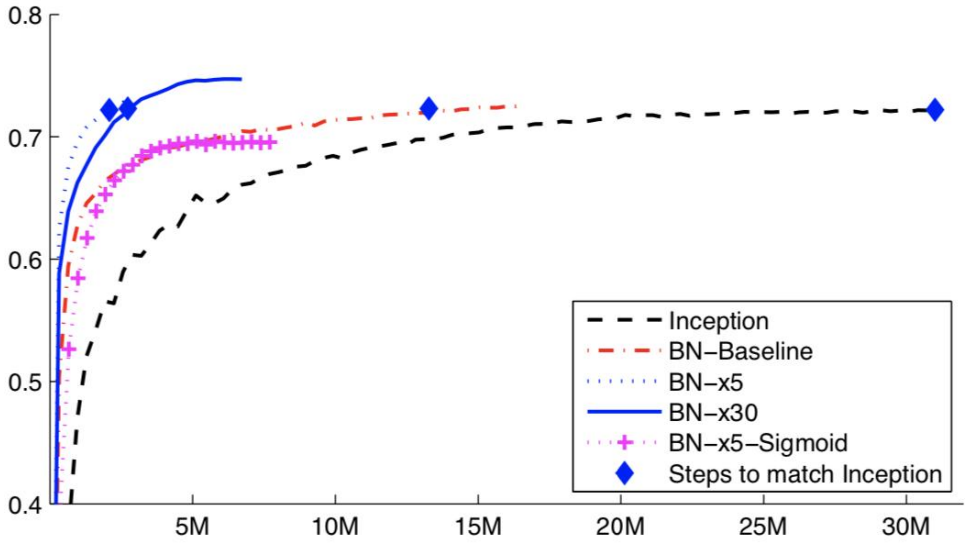


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

Features:

- Remove local response normalization (LRN) layers in previous architecture.
- GoogLeNet-V2 is trained faster than its GoogLeNet-V1 counterpart.
- ImageNet top-5 error: 4.9%

Sergey Ioffe et al., Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift (2015)

ResNet

Even with batch normalization, very deep neural networks are difficult to train.

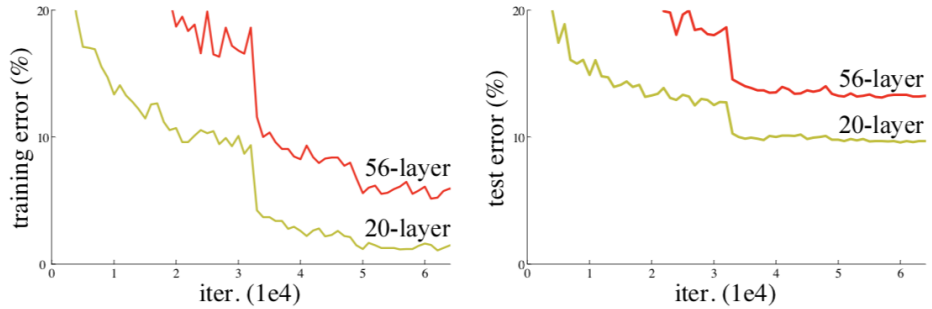


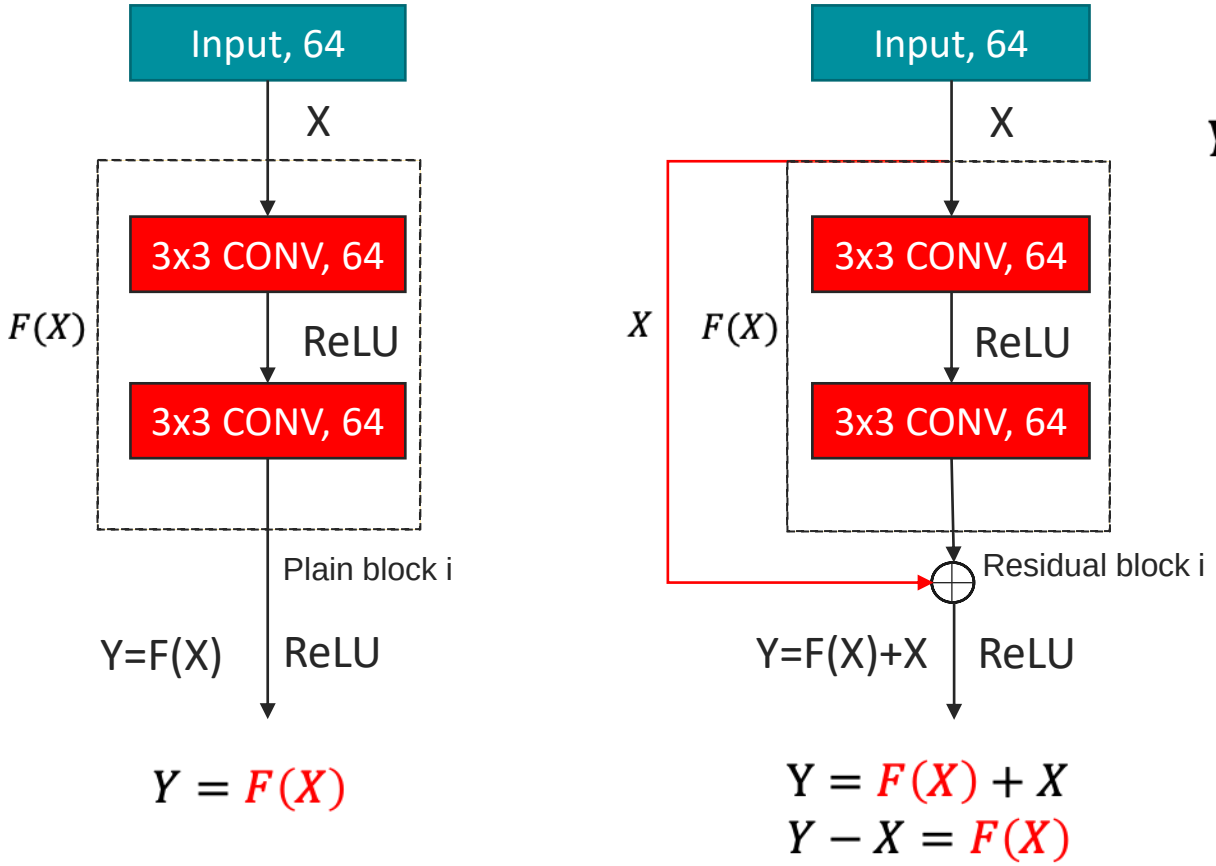
Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

- Optimization difficulty grows with the number of parameters in a deep neural network
- Gradient explosion/vanishing problems occur when the networks goes deeper.

He, Kaiming, et al. "Deep residual learning for image recognition." (2015)

ResNet

- During DNN training, we are trying to learn the function F(X)=Y. ResNet adds a shortcut connection to fit residual mapping F(X)=Y-X.



$$Y = F(x, \{W_i\}) + X$$

Residual mapping

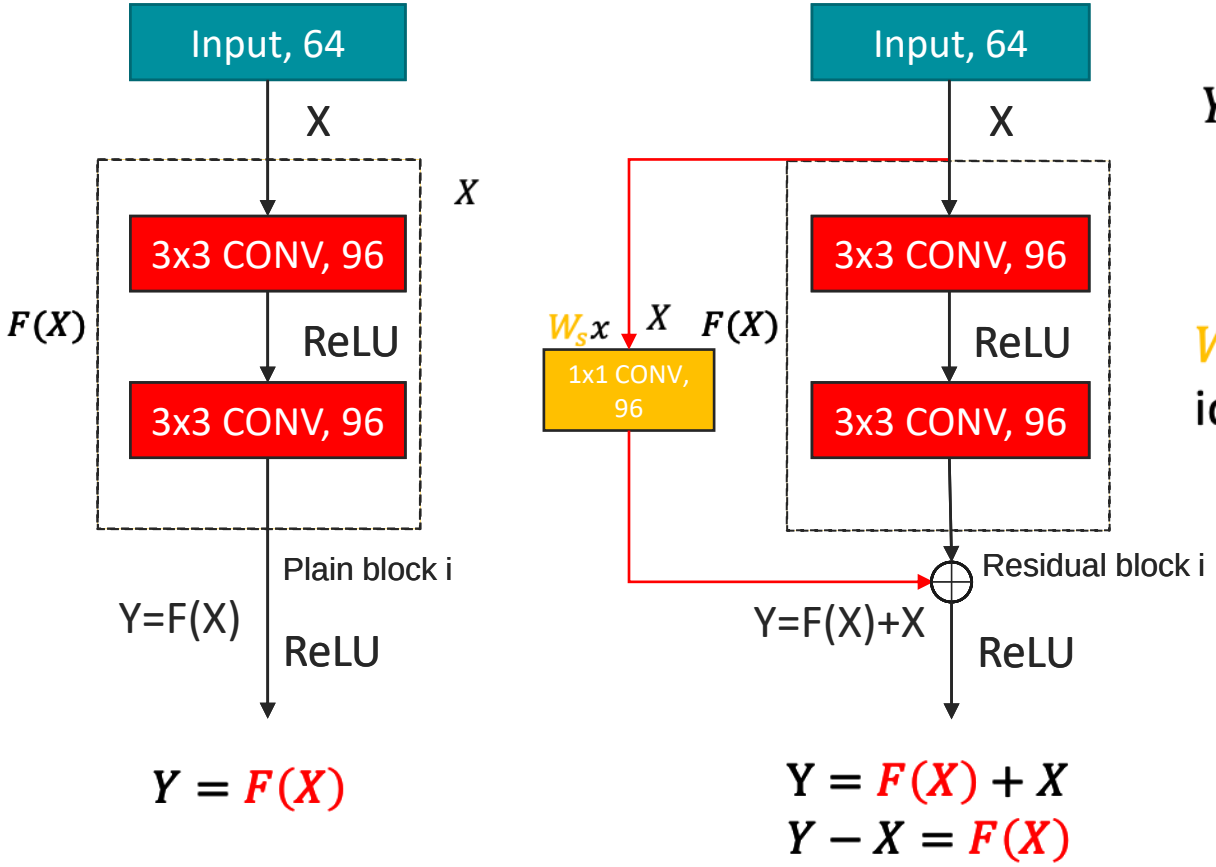
Identity mapping

He, Kaiming, et al. "Deep residual learning for image recognition." (2015)

ResNet

What if the number of output filters in Y changes after passing function F(X)?

•Solution: Use 1x1 convolution to match dimension.



$$Y = F(x, \{W_i\}) + W_s x$$

Original mapping — Residual mapping

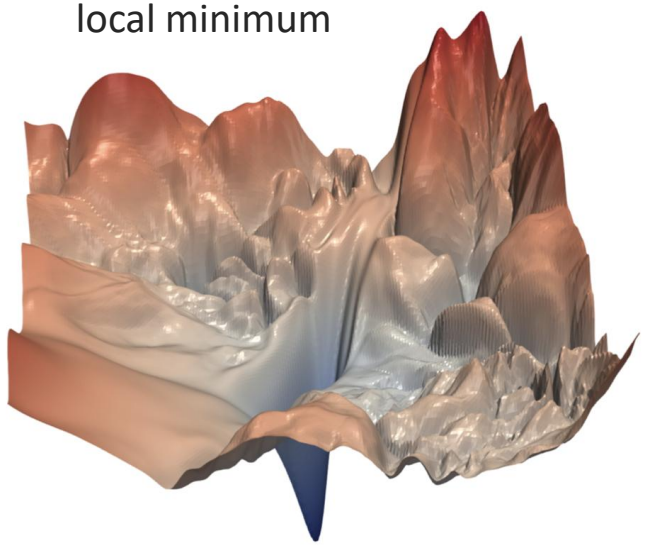
$W_s \in \mathrm{R}^{1\times1\times64\times96}$ is used for identity mapping.

ResNet

Visualization

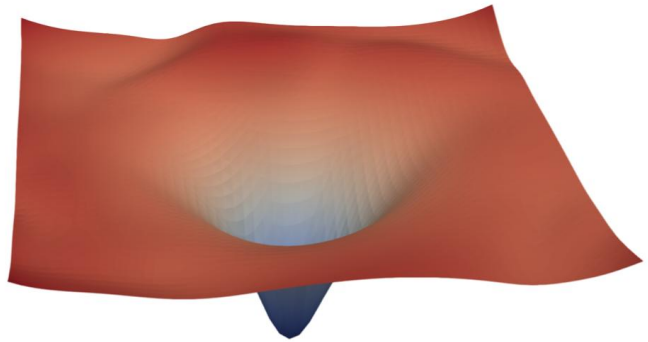
- How does residual learning simplify the optimization process?

Noisy, easy to get stuck into local minimum



Loss surface **without** shortcut connections.

Smooth & easy to optimize



Loss surface **with** shortcut connections.

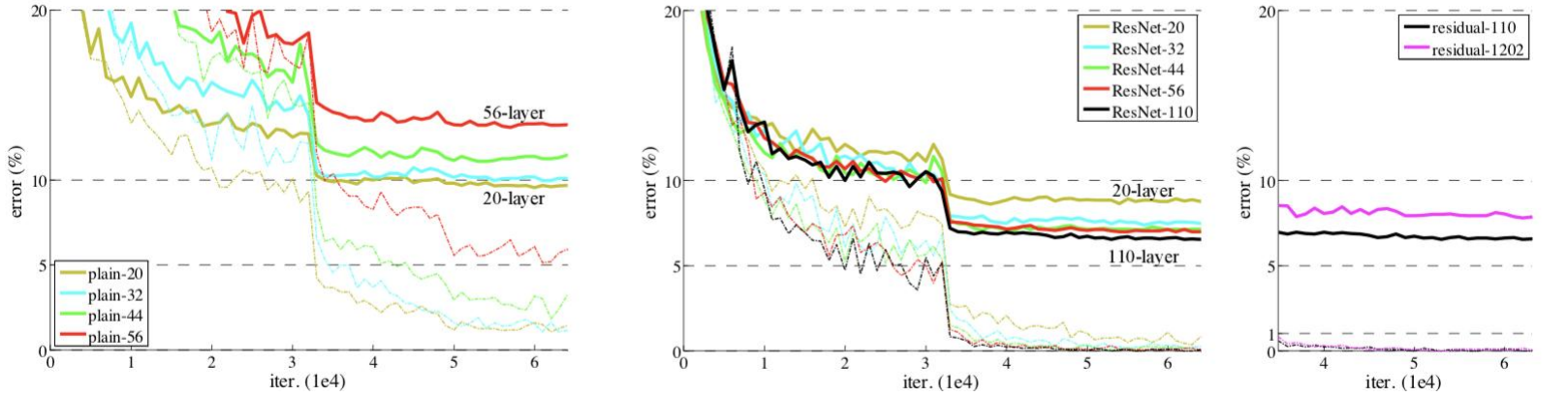
Shortcut connections make the loss surface smoother and easier to optimize.

Li Hao et al., Visualizing the Loss Landscape of Neural Nets

ResNet

Visualization

- How does loss value change in the ResNet training process?



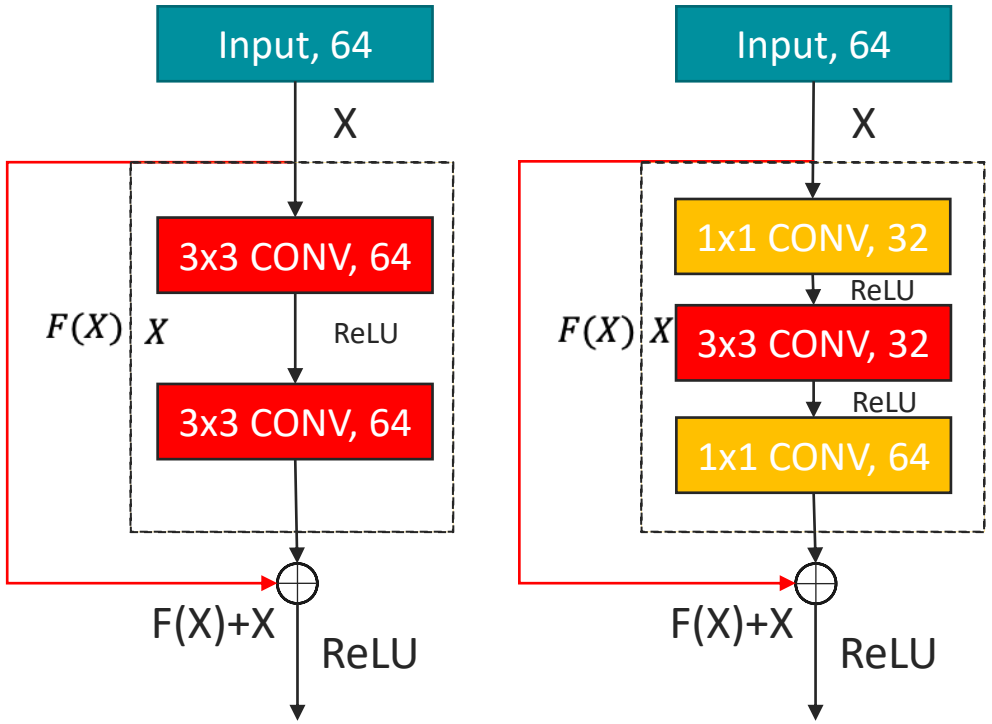
Plain networks **ResNets** **Very deep ResNets**

We cannot indefinitely add the depth of neural networks. ResNets with more than 1000 layers are still difficult to optimize!

ResNet

Deeper bottleneck architectures

- Similar to GoogLeNet, ResNet uses bottleneck layers to improve the efficiency of the model.



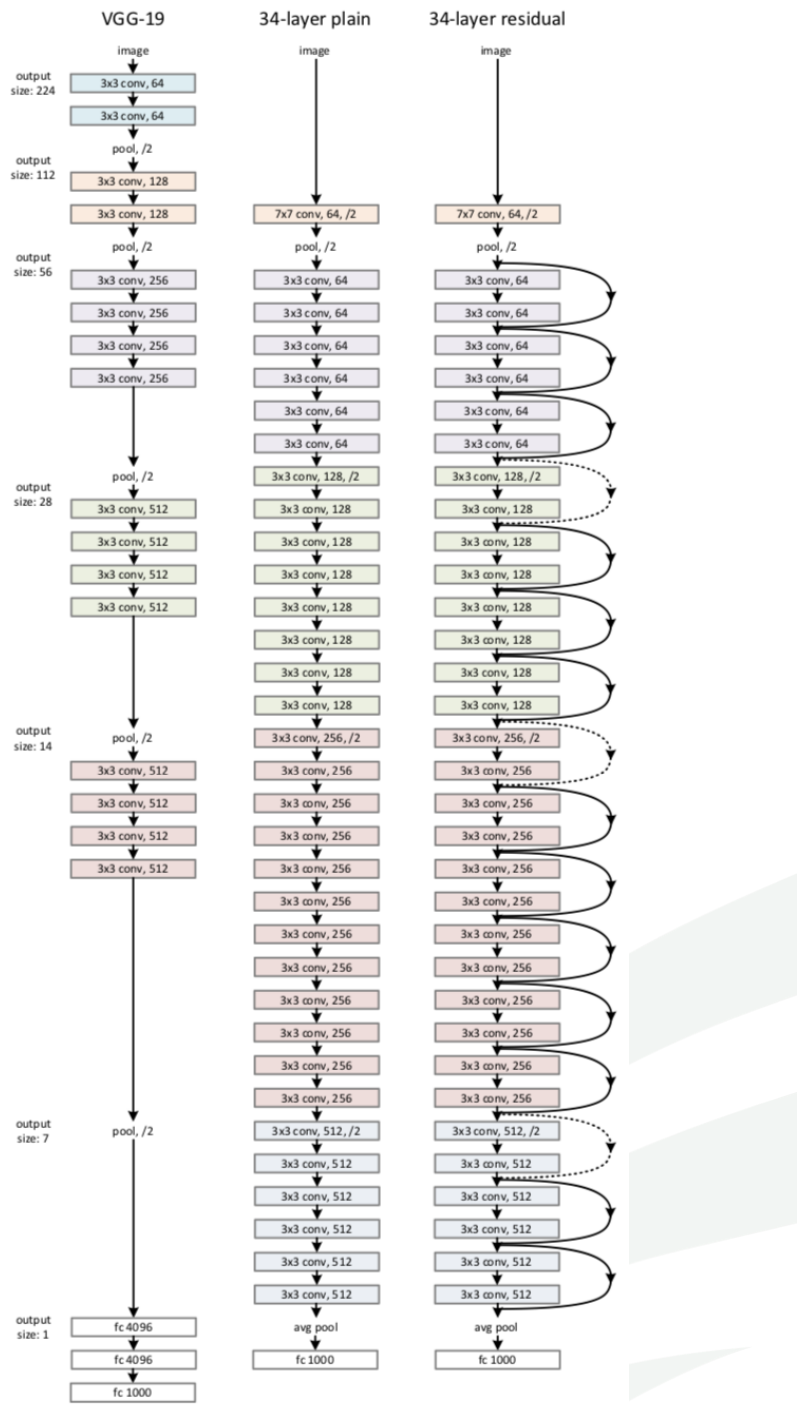
32 1x1 filters projects to 32 feature maps.

64 1x1 filters projects back to 64 feature maps.

ResNet

Summary

- ResNet constructs the neural network by stacking residual blocks. Each residual block has 2 3×3 convolutions.
- ResNet enables the design of very deep convolutional networks (from ResNet-18 to ResNet-152).
- By using bottleneck architectures, ResNet can be more efficient.
- ResNet achieves 3.57% top-5 error on ImageNet dataset.

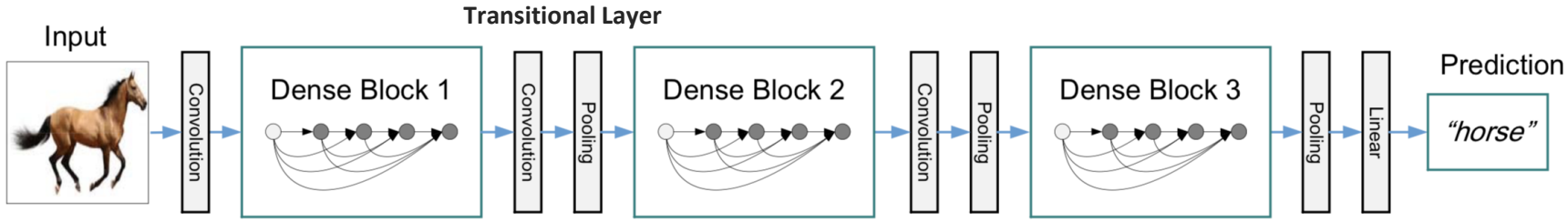


ResNet

- **How to train ResNet?**
  - Batch size is 256, use learning rate 0.01 to train for a few warmup epochs, then switch to initial learning rate 0.1.
  - Decay the learning rate by 0.1 when the validation accuracy plateaus.
  - Use SGD momentum with momentum 0.9
  - Use 1e-5 weight decay.
  - Use similar preprocessing methods as inception.

DenseNet

- DenseNet formulates dense connectivity between layers. Information can be passed with flexibility between different layers of DenseNets.



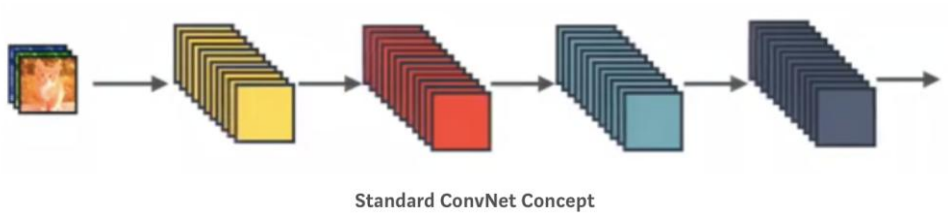
Features

- DenseNet encourages feature reuse and sharing between different layers.
- DenseNet uses transition layer to do down-sampling.
- With respect to feature map of similar depth, DenseNet reduces the weight parameter by using filter concatenation. However, DenseNet can lead to high memory consumption.

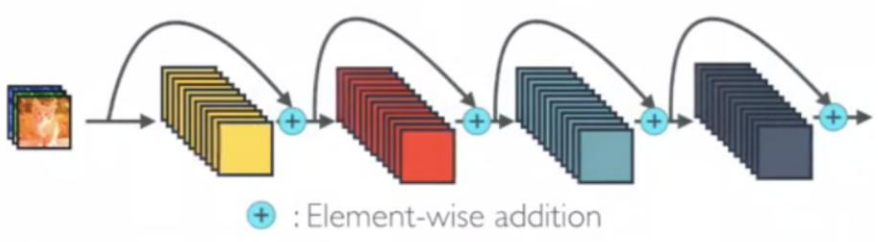
Huang, Gao, et al. "Densely connected convolutional networks." (2017)

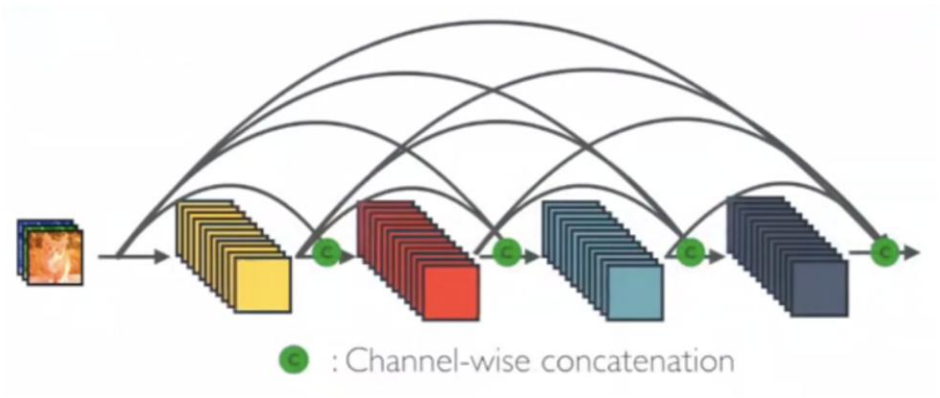
DenseNet

DenseNet filter concatenation



Standard ConvNet Concept





One Dense Block in DenseNet

Standard ConvNet:
Output of current layer is directly passed to the next layer.

ResNet:
Output of current layer is first added to the identity of the current residual block, then passed to the next layer.

DenseNet:
Output of the previous layers in the current dense block is first concatenated, and then passed to the next layer.

https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

DenseNet

- Training details:

  Similar to ResNet. (The authors adopted the same training methodology as ResNet for fair comparison.)

Summary of CNN architectures

Architecture	Single-Path?	Use small (<=3) convolutional kernel?	Depth?
AlexNet	✅	❌	Shallow (8 layers)
VGG	✅	✅	Deep (16-19 layers)
GoogLeNet	❌	✅(V3) ❌(V1,V2,V4)	Deeper (22 layers)
ResNet	❌	✅	Very deep (34-152 layers)
DenseNet	❌	✅	Very deep (121-264 layers)

Architecture	Input Size	Parameter Memory	MACs
AlexNet	224x224	233 MB	727 M
VGG-16/19	224x224	528 MB / 548 MB	16 G /20 G
GoogLeNet	224x224	51 MB	2 G
ResNet-18/34	224x224	45 MB / 83 MB	2 G / 4 G
DenseNet-121	224x224	31 MB	3 G

Summary of CNN architectures

Metrics	LeNet 5	AlexNet	Overfeat fast	VGG 16	GoogLeNet v1	ResNet 50
Top-5 error†	n/a	16.4	14.2	7.4	6.7	5.3
Top-5 error (single crop)†	n/a	19.8	17.0	8.8	10.7	7.0
Input Size	28×28	227×227	231×231	224×224	224×224	224×224
# of CONV Layers	2	5	5	13	57	53
Depth in # of CONV Layers	2	5	5	13	21	49
Filter Sizes	5	3,5,11	3,5,11	3	1,3,5,7	1,3,7
# of Channels	1, 20	3-256	3-1024	3-512	3-832	3-2048
# of Filters	20, 50	96-384	96-1024	64-512	16-384	64-2048
Stride	1	1,4	1,4	1	1,2	1,2
Weights	2.6k	2.3M	16M	14.7M	6.0M	23.5M
MACs	283k	666M	2.67G	15.3G	1.43G	3.86G
# of FC Layers	2	3	3	3	1	1
Filter Sizes	1,4	1,6	1,6,12	1,7	1	1
# of Channels	50, 500	256-4096	1024-4096	512-4096	1024	2048
# of Filters	10, 500	1000-4096	1000-4096	1000-4096	1000	1000
Weights	58k	58.6M	130M	124M	1M	2M
MACs	58k	58.6M	130M	124M	1M	2M
Total Weights	60k	61M	146M	138M	7M	25.5M
Total MACs	341k	724M	2.8G	15.5G	1.43G	3.9G
Pretrained Model Website	[56]‡	[57, 58]	n/a	[57–59]	[57–59]	[57–59]

https://arxiv.org/abs/1703.09039

Different deep neural nets

- Feedforward neural network (FNN)
  - –Output from one layer is used as input to the next layer
  - –No loops in the network - information is always fed forward, never fed back
- Recurrent neural networks (RNN)
  - –Has notion of dynamic change over time (time-varying behavior)
    - ▪Some neurons can fire for some limited duration of time and then go quiet
    - ▪That firing stimulate other neurons, which fire a little while later for a limited duration
    - ▪Over time we get a cascade of neurons firing
  - –Feedback loops are possible since a neuron's output only affects its input at some later time, not instantaneously
  - –There are many different ways to mathematically formalize our informal description of recurrent nets
  - –Learning algorithms for recurrent nets are (at least to date) less powerful
    - ▪Many ideas developed for FNN can also be used in RNN

Agenda

- Shallow vs deep neural networks
- Vanishing gradient problem for deep networks
- Deep convolutional neural networks (CNN)
- Max-pooling layer
- A practical CNN example
- The impact of GPU on deep learning
- ImageNet and Large-Scale Visual Recognition Challenge
- Different deep neural nets (DNN)
- Summary of recent techniques for DNN
- Conclusions

Quick summary of a few techniques that made training DNN possible

- Convolutional layers captures spatial information and greatly reduces the number of parameters
- Improved regularization techniques (e.g., dropout and pooling)
- Improved activation units (e.g., rectified linear instead of sigmoid)
- Using GPUs and being willing to train for a long period of time
- Making use of sufficiently large data sets

  Including algorithmically expanding the training data
- Using the right cost function and initialization (to avoid a learning slowdown)
- ……
- DNN is an active research field, but unfortunately, still lack of true theoretical breakthrough (mostly empirical + intuitive explanations)

Agenda

- Shallow vs deep neural networks
- Vanishing gradient problem for deep networks
- Deep convolutional neural networks (CNN)
- Max-pooling layer
- A practical CNN example
- The impact of GPU on deep learning
- ImageNet and Large-Scale Visual Recognition Challenge
- Different deep neural nets (DNN)
- Summary of recent techniques for DNN
- Conclusions

Conclusions

- Deep neural networks (deep nets) have become the dominant work horse for solving many interesting ML problems
- Convolutional neural network (CNN) is one of the most popular & well-studied DNN
- A number of key techniques have been developed in recent years to make training DNN possible
- The availability of computation power cannot be overemphasized enough

Lab 3: XXX

on Google Colab

Assignments and Related Documents:

- https://jqub.github.io/2021/09/01/ML4Emb/

Due Date: Next Friday (xx-xx-xxxx) by 1 PM

GMU.EDU

George Mason University

4400 University Drive
Fairfax, Virginia 22030

Tel: (703)993-1000