EE 628
Deep Learning
Fall 2019

Lecture 7

10/10/2019

Assistant Prof. Sergul Aydore

Department of Electrical and Computer Engineering

STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Overview

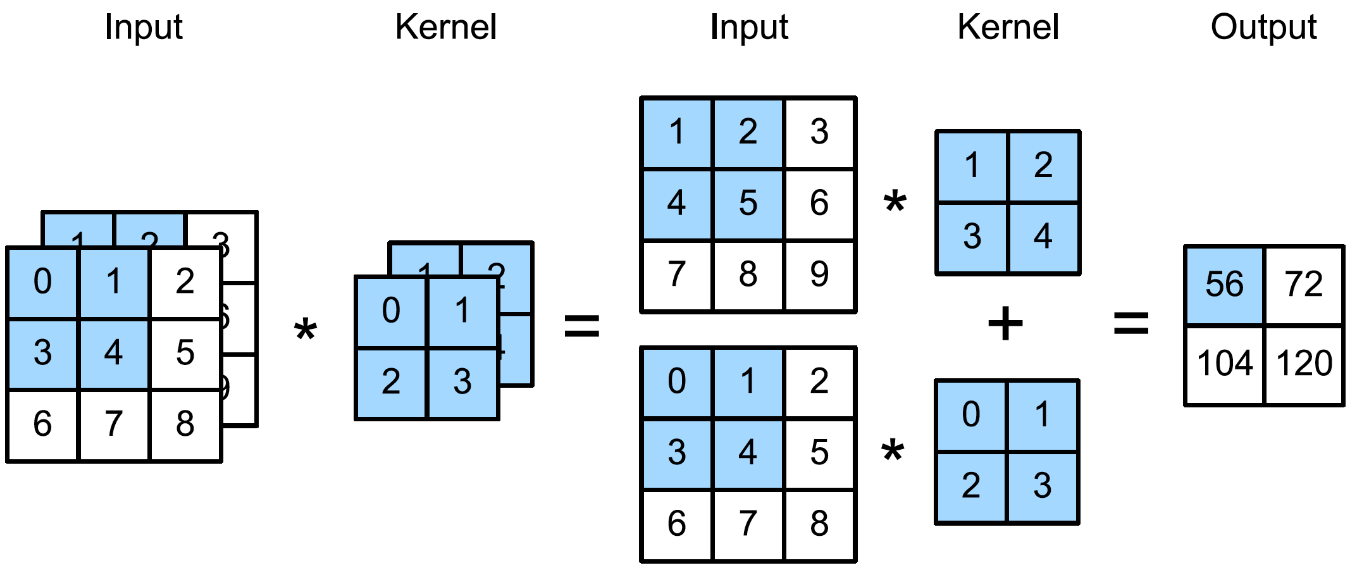
- Last lecture we covered
  - More on Optimization Algorithms
  - Convolutional layers
- Today, we will cover
  - More on convolutional layers
  - Convolutional neural networks

Multiple Input Channels

- When we add channels into the mix, our inputs and hidden representations both become three-dimensional arrays. For example, each RGB input image has shape 3 x h x w. We refer to this axis, with a size of 3, as the channel dimension.

Multiple Input Channels

- When we add channels into the mix, our inputs and hidden representations both become three-dimensional arrays. For example, each RGB input image has shape 3 x h x w. We refer to this axis, with a size of 3, as the channel dimension.
- When the input data contains multiple channels, we need to construct a convolution kernel with the same number of input channels as the input data



Multiple Output Channels

- Regardless of the number of input channels, so far we always ended up with one output channel.

Multiple Output Channels

- Regardless of the number of input channels, so far we always ended up with one output channel.
- In the most popular neural network architectures, we actually increase the channel dimension as we go higher up in the neural network.

Multiple Output Channels

- Regardless of the number of input channels, so far we always ended up with one output channel.
- In the most popular neural network architectures, we actually increase the channel dimension as we go higher up in the neural network.
- Intuitively, you could think of each channel as responding to some different set of features.

Multiple Output Channels

- Regardless of the number of input channels, so far we always ended up with one output channel.
- In the most popular neural network architectures, we actually increase the channel dimension as we go higher up in the neural network.
- Intuitively, you could think of each channel as responding to some different set of features.
- Denote by c_i and c_o the number of input and output channels, respectively, and let k_h and k_w be the height and width of the kernel.

Multiple Output Channels

- Regardless of the number of input channels, so far we always ended up with one output channel.
- In the most popular neural network architectures, we actually increase the channel dimension as we go higher up in the neural network.
- Intuitively, you could think of each channel as responding to some different set of features.
- Denote by c_i and c_o the number of input and output channels, respectively, and let k_h and k_w be the height and width of the kernel.
- To get an output with multiple channels, we can create a kernel array of shape $c_i \times k_h \times k_w$ for each output channel.

Multiple Output Channels

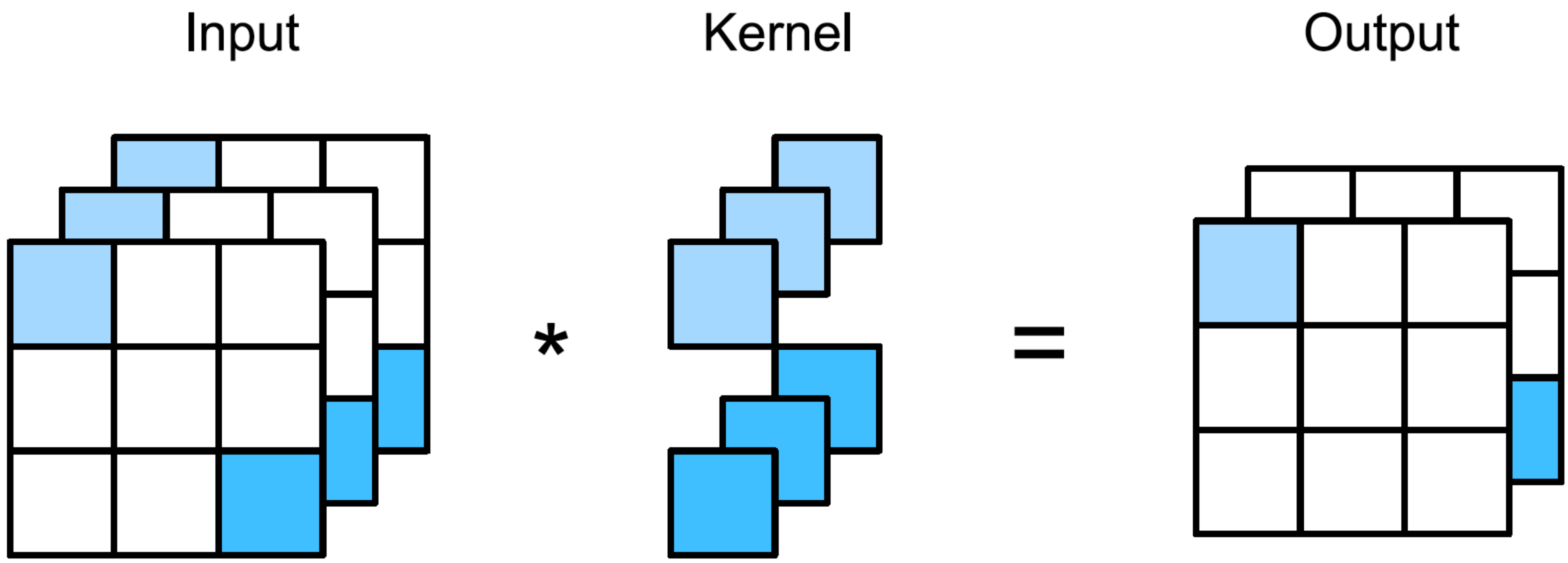
- Regardless of the number of input channels, so far we always ended up with one output channel.
- In the most popular neural network architectures, we actually increase the channel dimension as we go higher up in the neural network.
- Intuitively, you could think of each channel as responding to some different set of features.
- Denote by c_i and c_o the number of input and output channels, respectively, and let k_h and k_w be the height and width of the kernel.
- To get an output with multiple channels, we can create a kernel array of shape

$c_i \times k_h \times k_w$ for each output channel.

- We concatenate them on the output channel dimension, so that the shape of the convolution kernel is $c_o \times c_i \times k_h \times k_w$.

Multiple Input and Output Channels

- The figure below shows the cross-correlation computation using the 1 1 convolution kernel with 3 input channels and 2 output channels



Pooling

- Often, as we process images, we want to gradually reduce the spatial resolution of our hidden representations, aggregating information so that the higher up we go in the network, the larger the receptive field (in the input) to which each hidden node is sensitive.

Pooling

- Often, as we process images, we want to gradually reduce the spatial resolution of our hidden representations, aggregating information so that the higher up we go in the network, the larger the receptive field (in the input) to which each hidden node is sensitive.
- Pooling layers serve the dual purposes
  - of mitigating the sensitivity of convolutional layers to location and
  - of spatially downsampling representations

Maximum Pooling and Average Pooling

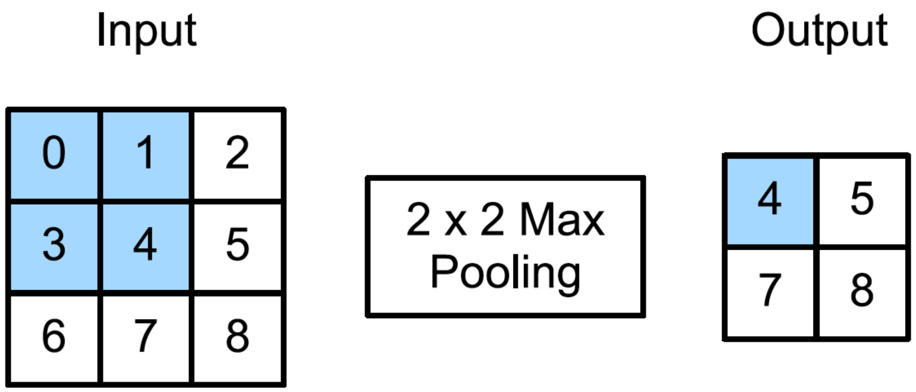
- Like convolutional layers, pooling operators consist of a fixed-shape window that slides over all regions in the input according to its stride, computing a single output for each location traversed by the fixed-shape window (sometimes known as the pooling window).

Maximum Pooling and Average Pooling

- Like convolutional layers, pooling operators consist of a fixed-shape window that slides over all regions in the input according to its stride, computing a single output for each location traversed by the fixed-shape window (sometimes known as the pooling window).
- However, the pooling layer contains no parameters.

Maximum Pooling and Average Pooling

- Like convolutional layers, pooling operators consist of a fixed-shape window that slides over all regions in the input according to its stride, computing a single output for each location traversed by the fixed-shape window (sometimes known as the pooling window).
- However, the pooling layer contains no parameters.
- Instead, pooling operators are deterministic, typically calculating either the maximum or the average value of the elements in the pooling window. These operations are called *maximum pooling* (max pooling for short) and *average pooling*, respectively.



Pooling Layers in Practice

- As with convolutional layers, pooling layers can also change the output shape.

Pooling Layers in Practice

- As with convolutional layers, pooling layers can also change the output shape.
- And as before, we can alter the operation to achieve a desired output shape by padding the input and adjusting the stride.

Pooling Layers in Practice

- As with convolutional layers, pooling layers can also change the output shape.
- And as before, we can alter the operation to achieve a desired output shape by padding the input and adjusting the stride.
- When processing multi-channel input data, the pooling layer pools each input channel separately, rather than adding the inputs of each channel by channel as in a convolutional layer.

Pooling Layers in Practice

- As with convolutional layers, pooling layers can also change the output shape.
- And as before, we can alter the operation to achieve a desired output shape by padding the input and adjusting the stride.
- When processing multi-channel input data, the pooling layer pools each input channel separately, rather than adding the inputs of each channel by channel as in a convolutional layer.
  - This means that the number of output channels for the pooling layer is the same as the number of input channels.

Convolutional Neural Networks (LeNet)

- In our first encounter with image data we applied a multilayer perceptron to Fashion-MNIST data set.

Convolutional Neural Networks (LeNet)

- In our first encounter with image data we applied a multilayer perceptron to Fashion-MNIST data set.
- Each image in Fashion-MNIST consisted of a two-dimensional 28 x 28 matrix.

Convolutional Neural Networks (LeNet)

- In our first encounter with image data we applied a multilayer perceptron to Fashion-MNIST data set.
- Each image in Fashion-MNIST consisted of a two-dimensional 28 x 28 matrix.
- We first flattened each image, yielding vectors of length 784, before processing them with a series of fully-connected layers.

Convolutional Neural Networks (LeNet)

- In our first encounter with image data we applied a multilayer perceptron to Fashion-MNIST data set.
- Each image in Fashion-MNIST consisted of a two-dimensional 28 x 28 matrix.
- We first flattened each image, yielding vectors of length 784, before processing them with a series of fully-connected layers.
- Now that we have introduced convolutional layers, we can keep the image in its original spatially-organized grid, processing it with a series of successive convolutional layers.

Convolutional Neural Networks (LeNet)

- In our first encounter with image data we applied a multilayer perceptron to Fashion-MNIST data set.
- Each image in Fashion-MNIST consisted of a two-dimensional 28 x 28 matrix.
- We first flattened each image, yielding vectors of length 784, before processing them with a series of fully-connected layers.
- Now that we have introduced convolutional layers, we can keep the image in its original spatially-organized grid, processing it with a series of successive convolutional layers.
- The first published convolutional neural networks whose benefit was

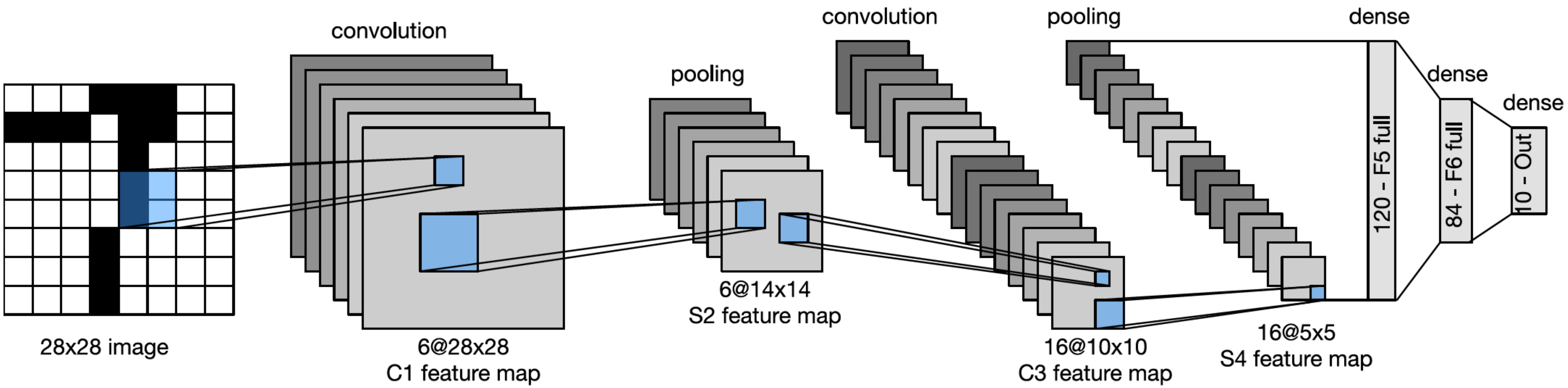
first demonstrated by Yann Lecun is LeNet5

LeNet

- We can think of LeNet as consisting two parts:
  - A block of convolutional layers; and
  - A block of fully-connected layers

LeNet

- We can think of LeNet as consisting two parts:
  - A block of convolutional layers; and
  - A block of fully-connected layers



LeNet

- Start IN_CLASS_convolutional_neural_networks.ipynb

Modern Convolutional Neural Networks

- We will cover neural network architectures that were at some point (or currently) the base model.

Modern Convolutional Neural Networks

- We will cover neural network architectures that were at some point (or currently) the base model.
- They were winners or runners-up in the famous ImageNet competition which has been the benchmark for supervised learning in computer vision since 2010.

Modern Convolutional Neural Networks

- We will cover neural network architectures that were at some point (or currently) the base model.
- They were winners or runners-up in the famous ImageNet competition which has been the benchmark for supervised learning in computer vision since 2010.
- These models include:
  - **Alexnet**: the first large-scale network deployed to beat conventional computer vision methods on a large-scale vision challenge.

Modern Convolutional Neural Networks

- We will cover neural network architectures that were at some point (or currently) the base model.
- They were winners or runners-up in the famous ImageNet competition which has been the benchmark for supervised learning in computer vision since 2010.
- These models include:
  - **Alexnet**: the first large-scale network deployed to beat conventional computer vision methods on a large-scale vision challenge.
  - The **VGG** network: makes use of a number of repeating blocks of elements.

Modern Convolutional Neural Networks

- We will cover neural network architectures that were at some point (or currently) the base model.
- They were winners or runners-up in the famous ImageNet competition which has been the benchmark for supervised learning in computer vision since 2010.
- These models include:
  - **Alexnet**: the first large-scale network deployed to beat conventional computer vision methods on a large-scale vision challenge.
  - The **VGG** network: makes use of a number of repeating blocks of elements.
  - The network in network (**NiN**): convolves whole neural networks patch-wise over inputs.

Modern Convolutional Neural Networks

- We will cover neural network architectures that were at some point (or currently) the base model.
- They were winners or runners-up in the famous ImageNet competition which has been the benchmark for supervised learning in computer vision since 2010.
- These models include:
  - **Alexnet**: the first large-scale network deployed to beat conventional computer vision methods on a large-scale vision challenge.
  - The **VGG** network: makes use of a number of repeating blocks of elements.
  - The network in network (**NiN**): convolves whole neural networks patch-wise over inputs.
  - The **GoogLeNet**: makes use of networks with parallel concentrations.

Modern Convolutional Neural Networks

- We will cover neural network architectures that were at some point (or currently) the base model.
- They were winners or runners-up in the famous ImageNet competition which has been the benchmark for supervised learning in computer vision since 2010.
- These models include:
  - **Alexnet**: the first large-scale network deployed to beat conventional computer vision methods on a large-scale vision challenge.
  - The **VGG** network: makes use of a number of repeating blocks of elements.
  - The network in network (**NiN**): convolves whole neural networks patch-wise over inputs.
  - The **GoogLeNet**: makes use of networks with parallel concentrations.
  - Residual Networks (**ResNet**): are currently the most popular go-to architecture today

Modern Convolutional Neural Networks

- We will cover neural network architectures that were at some point (or currently) the base model.
- They were winners or runners-up in the famous ImageNet competition which has been the benchmark for supervised learning in computer vision since 2010.
- These models include:
  - **Alexnet**: the first large-scale network deployed to beat conventional computer vision methods on a large-scale vision challenge.
  - The **VGG** network: makes use of a number of repeating blocks of elements.
  - The network in network (**NiN**): convolves whole neural networks patch-wise over inputs.
  - The **GoogLeNet**: makes use of networks with parallel concentrations.
  - Residual Networks (**ResNet**): are currently the most popular go-to architecture today
  - Densely connected networks (**DenseNet**): expensive to compute but have set some recent benchmarks.

Deep Convolutional Neural Networks (AlexNet)

- Classical pipelines looked more like this:
    1. Obtain an interesting dataset
    2. Preprocess the dataset with hand-crafted features based on some knowledge of optics, geometry, other analytic tools
    3. Feed the data through a standard set of feature extractions
    4. Dump the resulting representations into your favorite classifier, likely a linear model or kernel method, to learn a classifier
- CV researchers believed that a bigger or cleaner dataset or an improved feature-extraction pipeline mattered far more that any learning algorithm.

Deep Convolutional Neural Networks (AlexNet)

- Classical pipelines looked more like this:
  1. Obtain an interesting dataset
  2. Preprocess the dataset with hand-crafted features based on some knowledge of optics, geometry, other analytic tools
  3. Feed the data through a standard set of feature extractions
  4. Dump the resulting representations into your favorite classifier, likely a linear model or kernel method, to learn a classifier
- CV researchers believed that a bigger or cleaner dataset or an improved feature-extraction pipeline mattered far more that any learning algorithm.

Deep Convolutional Neural Networks (AlexNet)

- Classical pipelines looked more like this:

Deep Convolutional Neural Networks (AlexNet)

- Classical pipelines looked more like this:
    1. Obtain an interesting dataset

Deep Convolutional Neural Networks (AlexNet)

- Classical pipelines looked more like this:
  1. Obtain an interesting dataset
  2. Preprocess the dataset with hand-crafted features based on some knowledge of optics, geometry, other analytic tools

Deep Convolutional Neural Networks (AlexNet)

- Classical pipelines looked more like this:
  1. Obtain an interesting dataset
  2. Preprocess the dataset with hand-crafted features based on some knowledge of optics, geometry, other analytic tools
  3. Feed the data through a standard set of feature extractions

Deep Convolutional Neural Networks (AlexNet)

- Classical pipelines looked more like this:
  1. Obtain an interesting dataset
  2. Preprocess the dataset with hand-crafted features based on some knowledge of optics, geometry, other analytic tools
  3. Feed the data through a standard set of feature extractions
  4. Dump the resulting representations into your favorite classifier, likely a

Deep Convolutional Neural Networks (AlexNet)

- Classical pipelines looked more like this:
  1. Obtain an interesting dataset
  2. Preprocess the dataset with hand-crafted features based on some knowledge of optics, geometry, other analytic tools
  3. Feed the data through a standard set of feature extractions
  4. Dump the resulting representations into your favorite classifier, likely a linear model or kernel method, to learn a classifier
- CV researchers believed that a bigger or cleaner dataset or an improved feature-extraction pipeline mattered far more that any learning algorithm.

Learning Feature Representation

- So the most important part of the pipeline was the representation.

Learning Feature Representation

- So the most important part of the pipeline was the representation.
- Up until 2012 the representation was calculated mechanically.
- Another group of researchers, including Yann Lecun, Geoff Hinton, Yoshua Bengio, Andrew Ng, had different plans.
- They believed that features themselves should be learned.
- Moreover, they believed that to be reasonably complex, the features should be hierarchically composed with multiple layers

Learning Feature Representation

- So the most important part of the pipeline was the representation.
- Up until 2012 the representation was calculated mechanically.

Learning Feature Representation

- So the most important part of the pipeline was the representation.
- Up until 2012 the representation was calculated mechanically.
- Another group of researchers, including Yann Lecun, Geoff Hinton, Yoshua Bengio, Andrew Ng, had different plans.

Learning Feature Representation

- So the most important part of the pipeline was the representation.
- Up until 2012 the representation was calculated mechanically.
- Another group of researchers, including Yann Lecun, Geoff Hinton, Yoshua Bengio, Andrew Ng, had different plans.
- They believed that features themselves should be learned.

Learning Feature Representation

- So the most important part of the pipeline was the representation.
- Up until 2012 the representation was calculated mechanically.
- Another group of researchers, including Yann Lecun, Geoff Hinton, Yoshua Bengio, Andrew Ng, had different plans.
- They believed that features themselves should be learned.
- Moreover, they believed that to be reasonably complex, the features should be hierarchically composed with multiple layers

Learning Feature Representation

- In the case of an image, the lowest layers might come to detect edges, colors, and textures.

Learning Feature Representation

- In the case of an image, the lowest layers might come to detect edges, colors, and textures.
- Higher layers in the network might build upon these representations to represent larger structures, like eyes, noses, etc.

Learning Feature Representation

- In the case of an image, the lowest layers might come to detect edges, colors, and textures.
- Higher layers in the network might build upon these representations to represent larger structures, like eyes, noses, etc.
- Even higher layers in the network might represent whole objects like people, airplanes, dogs, etc.

Learning Feature Representation

- In the case of an image, the lowest layers might come to detect edges, colors, and textures.
- Higher layers in the network might build upon these representations to represent larger structures, like eyes, noses, etc.
- Even higher layers in the network might represent whole objects like people, airplanes, dogs, etc.
- Ultimately, the final hidden state learns a compact representation of the image that summarizes its contents such that the data belonging to different categories can be separated easily.

Image filters learned by the first layer of AlexNet

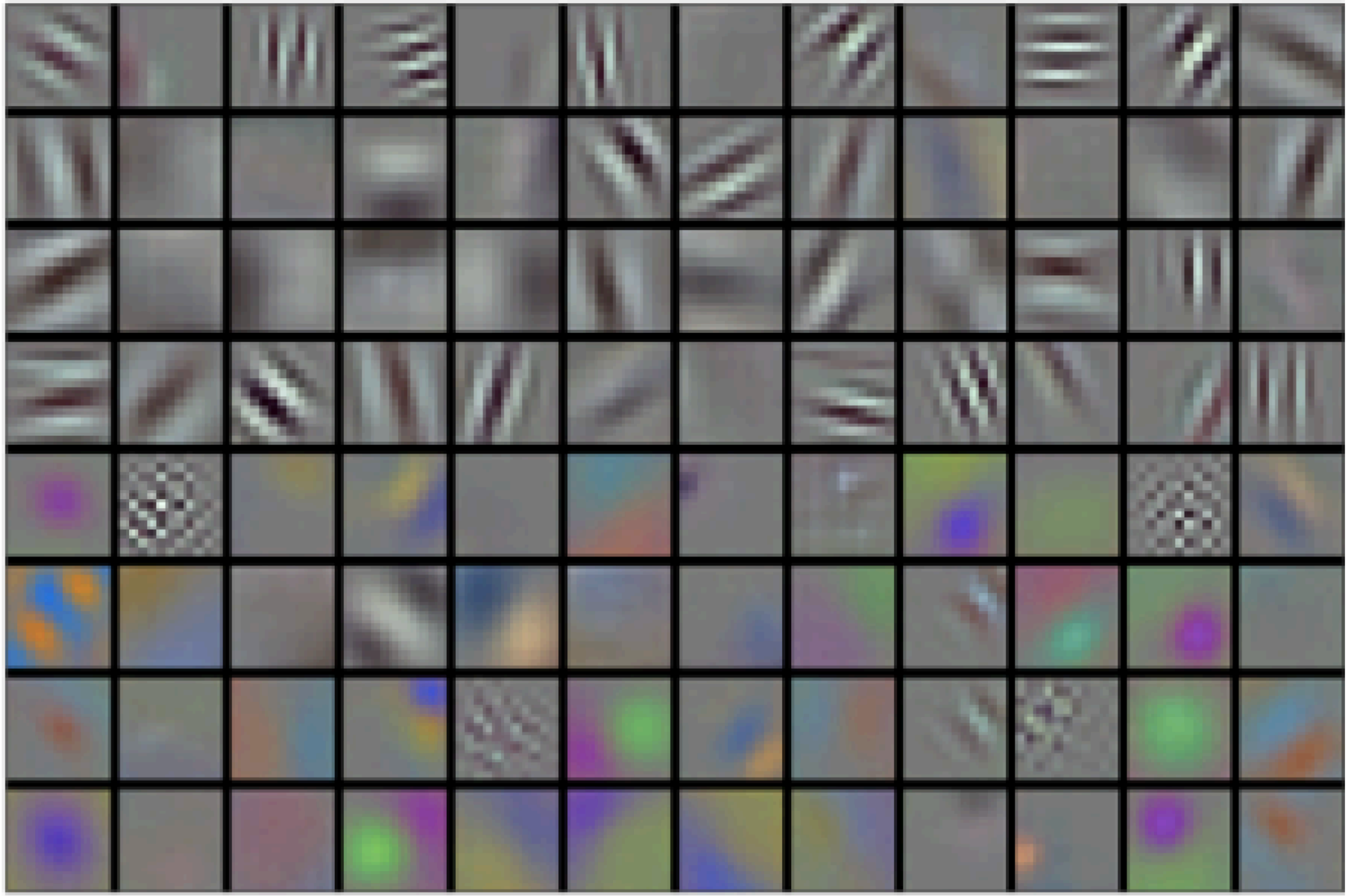


Fig. 9.1.1: Image filters learned by the first layer of AlexNet

Key factors for success of the breakthrough in 2012

- Missing Ingredient – Data:

Key factors for success of the breakthrough in 2012

- Missing Ingredient – Data:
  - Most research relied on tiny datasets because of limited storage capacity of computers, the relative expense of sensors, and the comparatively tighter research budgets in 1990s.

Key factors for success of the breakthrough in 2012

- Missing Ingredient – Data:
  - Most research relied on tiny datasets because of limited storage capacity of computers, the relative expense of sensors, and the comparatively tighter research budgets in 1990s.
  - In 2009. the ImageNet dataset was released, challenging researchers to learn from 1 million examples, 1000 each from 1000 distinct categories of objects.

Key factors for success of the breakthrough in 2012

- Missing Ingredient – Data:
    - Most research relied on tiny datasets because of limited storage capacity of computers, the relative expense of sensors, and the comparatively tighter research budgets in 1990s.
    - In 2009. the ImageNet dataset was released, challenging researchers to learn from 1 million examples, 1000 each from 1000 distinct categories of objects.
    - ImageNet challenge pushed computer vision and ML research forward by challenging researchers to identify which models performed best at a greater scale.

Key factors for success of the breakthrough in 2012

- Missing Ingredient – Data:
  - Most research relied on tiny datasets because of limited storage capacity of computers, the relative expense of sensors, and the comparatively tighter research budgets in 1990s.
  - In 2009. the ImageNet dataset was released, challenging researchers to learn from 1 million examples, 1000 each from 1000 distinct categories of objects.
  - ImageNet challenge pushed computer vision and ML research forward by challenging researchers to identify which models performed best at a greater scale.
- Missing Ingredients – Hardware:

Key factors for success of the breakthrough in 2012

- Missing Ingredient – Data:
  - Most research relied on tiny datasets because of limited storage capacity of computers, the relative expense of sensors, and the comparatively tigher research budgets in 1990s.
  - In 2009. the ImageNet dataset was released, challenging researchers to learn from 1 million examples, 1000 each from 1000 distinct categories of objects.
  - ImageNet challenge pushed computer vision and ML research forward by challenging researchers to identify which models performed best at a greater scale.
- Missing Ingredients – Hardware:
  - DL models demand expensive computations
  - GPUs proved to be a game changer

AlexNet

- Introduced in 2012, named after Alex Krizhevsky, the first author of the breakthrough ImageNet classification paper [ImageNet Classification with Deep Neural Networks, NIPS 2012]

AlexNet

- Introduced in 2012, named after Alex Krizhevsky, the first author of the breakthrough ImageNet classification paper [ImageNet Classification with Deep Neural Networks, NIPS 2012]
- AlexNet won the ImageNet Large Scale Visual Recognition challenge 2012 by a phenomenally large margin.

AlexNet

- Introduced in 2012, named after Alex Krizhevsky, the first author of the breakthrough ImageNet classification paper [ImageNet Classification with Deep Neural Networks, NIPS 2012]
- AlexNet won the ImageNet Large Scale Visual Recognition challenge 2012 by a phenomenally large margin.
- This network proved, for the first time, that the features obtained by learning can transcend manually-design features, breaking the previous paradigm in computer vision.

AlexNet

- Introduced in 2012, named after Alex Krizhevsky, the first author of the breakthrough ImageNet classification paper [ImageNet Classification with Deep Neural Networks, NIPS 2012]
- AlexNet won the ImageNet Large Scale Visual Recognition challenge 2012 by a phenomenally large margin.
- This network proved, for the first time, that the features obtained by learning can transcend manually-design features, breaking the previous paradigm in computer vision.
- The design philosophies of AlexNet and LeNet are very similar, but there are some significant differences.

AlexNet

- Introduced in 2012, named after Alex Krizhevsky, the first author of the breakthrough ImageNet classification paper [ImageNet Classification with Deep Neural Networks, NIPS 2012]
- AlexNet won the ImageNet Large Scale Visual Recognition challenge 2012 by a phenomenally large margin.
- This network proved, for the first time, that the features obtained by learning can transcend manually-design features, breaking the previous paradigm in computer vision.
- The design philosophies of AlexNet and LeNet are very similar, but there are some significant differences.
  - AlexNet is much deeper than the comparatively small LeNet-5.

AlexNet

- Introduced in 2012, named after Alex Krizhevsky, the first author of the breakthrough ImageNet classification paper [ImageNet Classification with Deep Neural Networks, NIPS 2012]
- AlexNet won the ImageNet Large Scale Visual Recognition challenge 2012 by a phenomenally large margin.
- This network proved, for the first time, that the features obtained by learning can transcend manually-design features, breaking the previous paradigm in computer vision.
- The design philosophies of AlexNet and LeNet are very similar, but there are some significant differences.
  - AlexNet is much deeper than the comparatively small LeNet-5.
  - AlexNet used the ReLU instead of the sigmoid as its activation function.

AlexNet - Architecture

- Compared to LeNet, AlexNet
  - has larger convolution window,
  - has three more convolution layers,
  - adds max pooling layers and
  - has ten times more convolution channels.

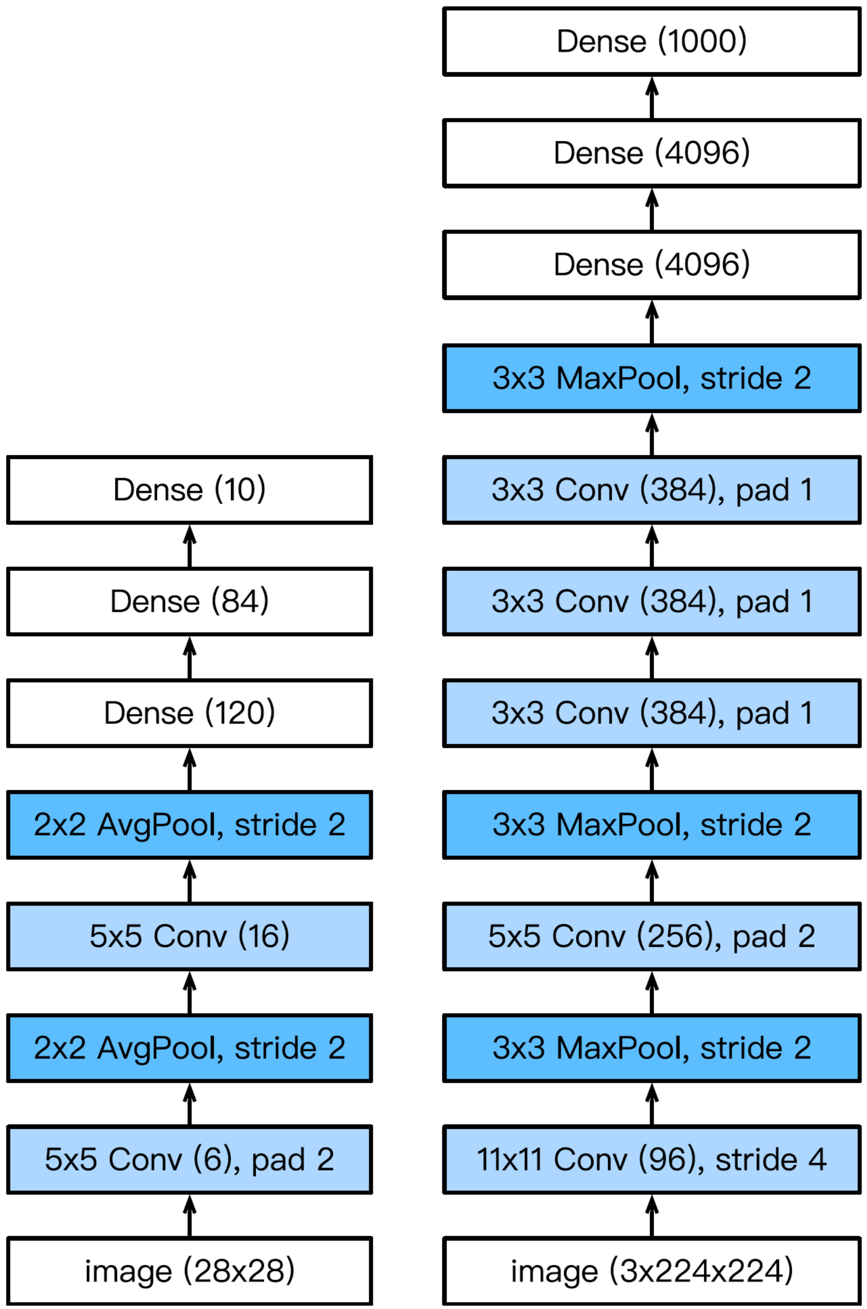


Fig. 9.1.2: LeNet (left) and AlexNet (right)

AlexNet - Architecture

- Compared to LeNet, AlexNet
  - has larger convolution window,
  - has three more convolution layers,
  - adds max pooling layers and
  - has ten times more convolution channels.
- Due to the limited memory in early GPUs, the original AlexNet used dual data stream design, so that each of their two GPUs could be responsible for storing and computing only its half of the model.

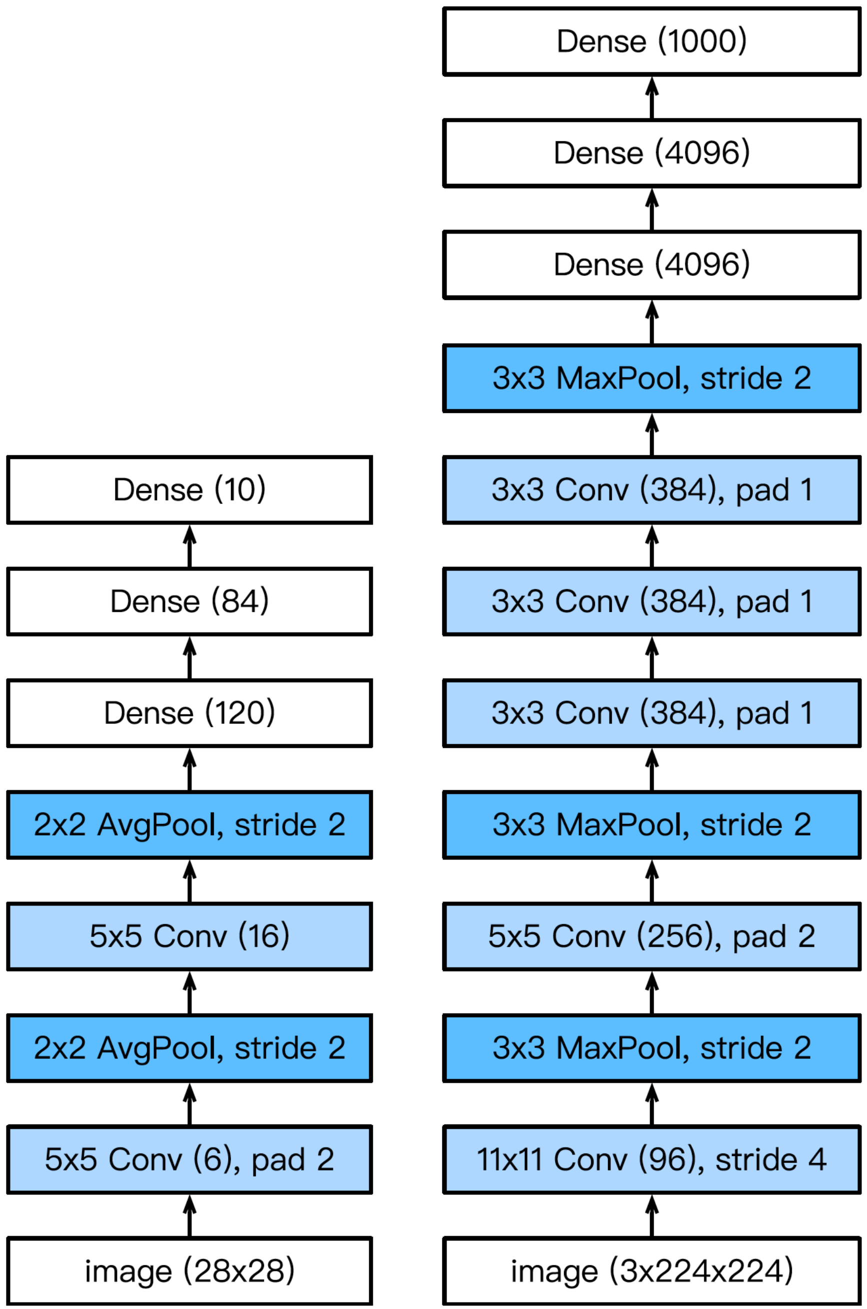


Fig. 9.1.2: LeNet (left) and AlexNet (right)

AlexNet - Architecture

- Compared to LeNet, AlexNet
  - has larger convolution window,
  - has three more convolution layers,
  - adds max pooling layers and
  - has ten times more convolution channels.
- Due to the limited memory in early GPUs, the original AlexNet used dual data stream design, so that each of their two GPUs could be responsible for storing and computing only its half of the model.
- Fortunately, GPU memory is comparatively abundant now, so we rarely need to break up models across GPUs these days.

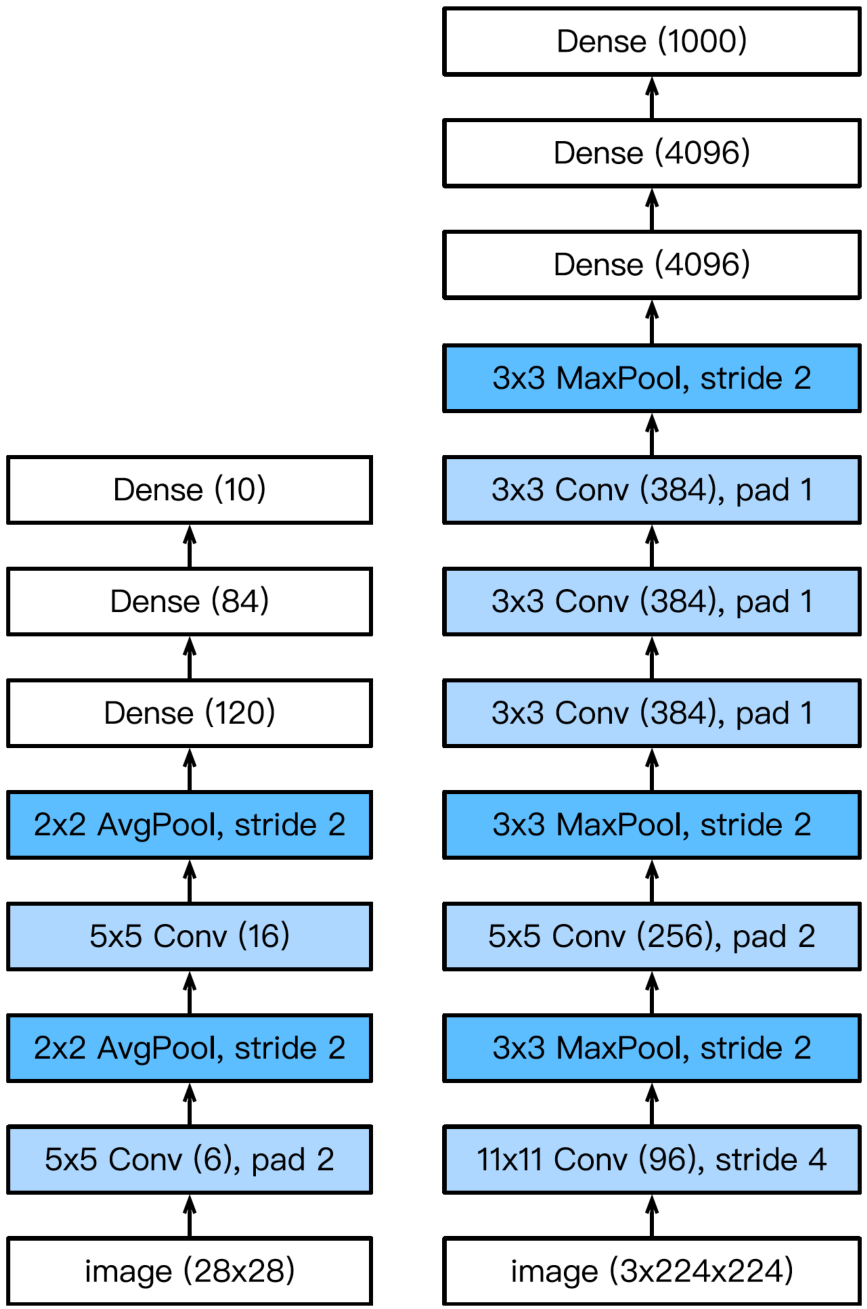


Fig. 9.1.2: LeNet (left) and AlexNet (right)

AlexNet – Activation Functions

- AlexNet changed the sigmoid activation function to a simpler ReLU activation function.

AlexNet – Activation Functions

- AlexNet changed the sigmoid activation function to a simpler ReLU activation function.
- The computation of the ReLU activation function is simpler. It does not have the exponentiation operation found in the sigmoid activation function.

AlexNet – Activation Functions

- AlexNet changed the sigmoid activation function to a simpler ReLU activation function.
- The computation of the ReLU activation function is simpler. It does not have the exponentiation operation found in the sigmoid activation function.
- ReLU makes model training easier when using different parameter initialization methods, this is because:

AlexNet – Activation Functions

- AlexNet changed the sigmoid activation function to a simpler ReLU activation function.
- The computation of the ReLU activation function is simpler. It does not have the exponentiation operation found in the sigmoid activation function.
- ReLU makes model training easier when using different parameter initialization methods, this is because:
  - The gradient of the ReLU activation function in the positive interval is always 1.

AlexNet – Activation Functions

- AlexNet changed the sigmoid activation function to a simpler ReLU activation function.
- The computation of the ReLU activation function is simpler. It does not have the exponentiation operation found in the sigmoid activation function.
- ReLU makes model training easier when using different parameter initialization methods, this is because:
  - The gradient of the ReLU activation function in the positive interval is always 1.
  - However, if the model parameters are not initialized properly, the sigmoid function may obtain a gradient of almost 0 so that the model cannot be effectively trained.

AlexNet – Capacity Control and Preprocessing

- AlexNet controls the model complexity of the fully-connected layer by dropout while LeNet uses weight decay.

AlexNet – Capacity Control and Preprocessing

- AlexNet controls the model complexity of the fully-connected layer by dropout while LeNet uses weight decay.
- The training loop of AlexNet added a great deal of image augmentation, such as flipping, clipping and color changes.

AlexNet – Capacity Control and Preprocessing

- AlexNet controls the model complexity of the fully-connected layer by dropout while LeNet uses weight decay.
- The training loop of AlexNet added a great deal of image augmentation, such as flipping, clipping and color changes.
- This makes the model more robust and the larger sample size effectively reduces overfitting.

Networks Using Blocks

- While AlexNet proved that deep convolutional neural networks can achieve good results, it didn’t offer a general template to guide subsequent researchers in designing new networks.

Networks Using Blocks

- While AlexNet proved that deep convolutional neural networks can achieve good results, it didn’t offer a general template to guide subsequent researchers in designing new networks.
- The design of neural network architectures had grown with researchers moving from thinking in terms of individual neurons to whole layers, and now to blocks, repeating patterns of layers.

Networks Using Blocks

- While AlexNet proved that deep convolutional neural networks can achieve good results, it didn’t offer a general template to guide subsequent researchers in designing new networks.
- The design of neural network architectures had grown with researchers moving from thinking in terms of individual neurons to whole layers, and now to blocks, repeating patterns of layers.
- The idea of using blocks first emerged from the Visual Geometry Group120 (VGG) at Oxford University.

Networks Using Blocks

- While AlexNet proved that deep convolutional neural networks can achieve good results, it didn’t offer a general template to guide subsequent researchers in designing new networks.
- The design of neural network architectures had grown with researchers moving from thinking in terms of individual neurons to whole layers, and now to blocks, repeating patterns of layers.
- The idea of using blocks first emerged from the Visual Geometry Group120 (VGG) at Oxford University.
- In their VGG network, it’s easy to implement these repeated structures in code with any modern deep learning framework by using loops and subroutines.

VGG Blocks

- The basic building block of classic convolutional networks is a sequence of the following layers:

VGG Blocks

- The basic building block of classic convolutional networks is a sequence of the following layers:
  - a convolutional layer (with padding to maintain the resolution)

VGG Blocks

- The basic building block of classic convolutional networks is a sequence of the following layers:
  - a convolutional layer (with padding to maintain the resolution)
  - a nonlinearity such as a ReLu

VGG Blocks

- The basic building block of classic convolutional networks is a sequence of the following layers:
  - a convolutional layer (with padding to maintain the resolution)
  - a nonlinearity such as a ReLu
- One VGG block consists of a sequence of convolutional layers, followed by a max pooling layer for spatial downsampling.

VGG Network

- Like AlexNet and LeNet, the VGG Network can be partitioned into two parts: the first consisting mostly of convolutional and pooling layers and a second consisting of fully-connected layers.

VGG Network

- Like AlexNet and LeNet, the VGG Network can be partitioned into two parts: the first consisting mostly of convolutional and pooling layers and a second consisting of fully-connected layers.
- The convolutional portion of the net connects several vgg blocks in succession.

VGG Network

- Like AlexNet and LeNet, the VGG Network can be partitioned into two parts: the first consisting mostly of convolutional and pooling layers and a second consisting of fully-connected layers.
- The convolutional portion of the net connects several vgg blocks in succession.
- The original VGG network had 5 convolutional blocks, among which the first two have one convolutional layer each and the latter three contain two convolutional layers each. [paper: https://arxiv.org/pdf/1409.1556.pdf]

VGG Network

- Like AlexNet and LeNet, the VGG Network can be partitioned into two parts: the first consisting mostly of convolutional and pooling layers and a second consisting of fully-connected layers.
- The convolutional portion of the net connects several vgg blocks in succession.
- The original VGG network had 5 convolutional blocks, among which the first two have one convolutional layer each and the latter three contain two convolutional layers each. [paper: https://arxiv.org/pdf/1409.1556.pdf]
- Since this network uses 8 convolutional layers and 3 fully-connected layers, it is often called VGG-11.

VGG Network

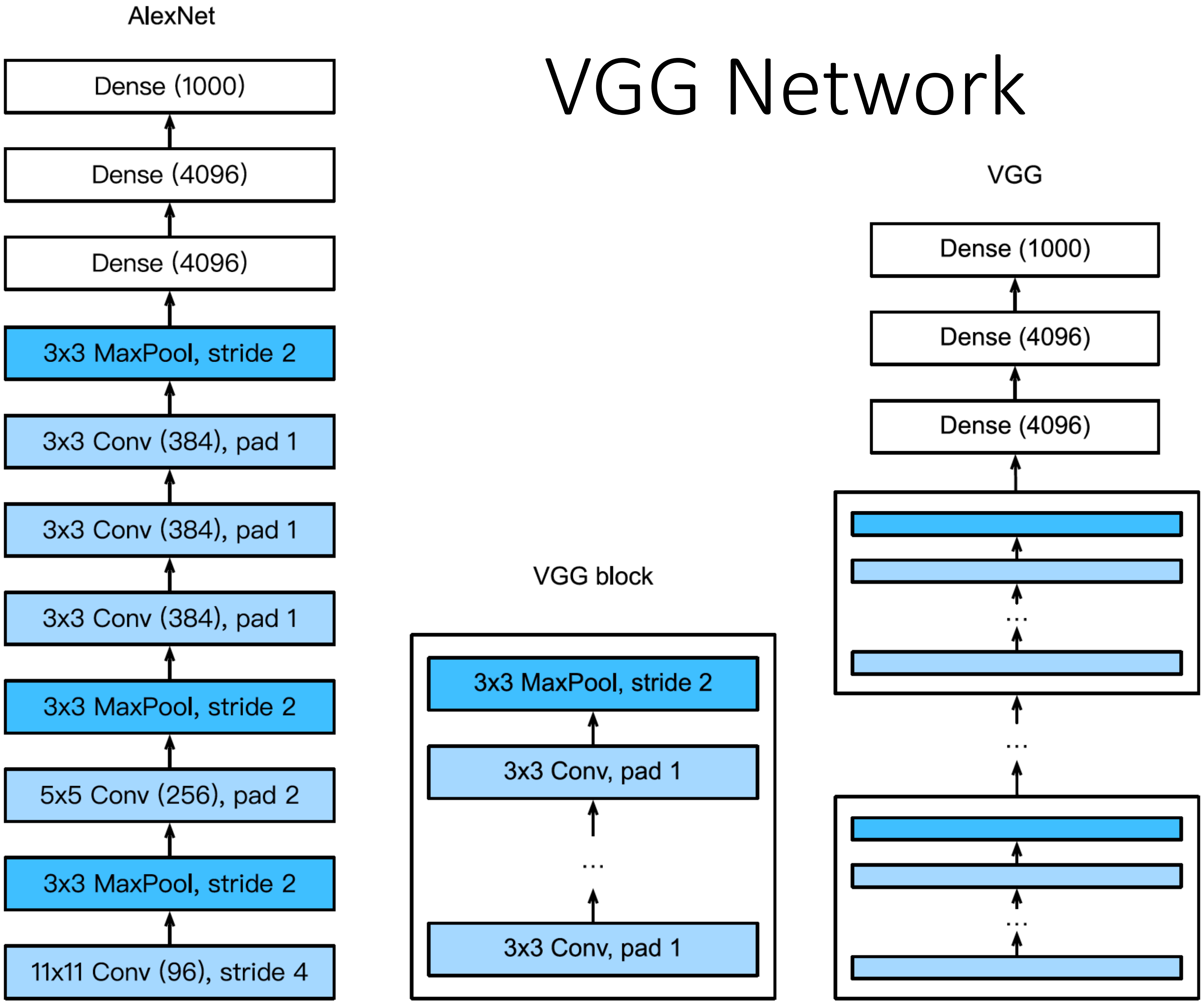


Fig. 9.2.1: Designing a network from building blocks

Network in Network (NiN)

- LeNet, AlexNet, and VGG all share a common design pattern: extract features exploiting spatial structure via a sequence of convolutions and pooling layers and then post-process the representations via fully-connected layers.

Network in Network (NiN)

- LeNet, AlexNet, and VGG all share a common design pattern: extract features exploiting spatial structure via a sequence of convolutions and pooling layers and then post-process the representations via fully-connected layers.
- The improvements upon LeNet by AlexNet and VGG mainly lie in how these later networks widen and deepen these two modules.

Network in Network (NiN)

- LeNet, AlexNet, and VGG all share a common design pattern: extract features exploiting spatial structure via a sequence of convolutions and pooling layers and then post-process the representations via fully-connected layers.
- The improvements upon LeNet by AlexNet and VGG mainly lie in how these later networks widen and deepen these two modules.
- Alternatively, one could imagine using fully-connected layers earlier in the process.

Network in Network (NiN)

- LeNet, AlexNet, and VGG all share a common design pattern: extract features exploiting spatial structure via a sequence of convolutions and pooling layers and then post-process the representations via fully-connected layers.
- The improvements upon LeNet by AlexNet and VGG mainly lie in how these later networks widen and deepen these two modules.
- Alternatively, one could imagine using fully-connected layers earlier in the process.
- However, a careless use of dense layers might give up the spatial structure of the representation entirely, Network in Network (NiN) blocks offer an alternative.

Network in Network (NiN)

- LeNet, AlexNet, and VGG all share a common design pattern: extract features exploiting spatial structure via a sequence of convolutions and pooling layers and then post-process the representations via fully-connected layers.
- The improvements upon LeNet by AlexNet and VGG mainly lie in how these later networks widen and deepen these two modules.
- Alternatively, one could imagine using fully-connected layers earlier in the process.
- However, a careless use of dense layers might give up the spatial structure of the representation entirely, Network in Network (NiN) blocks offer an alternative.
- NiN blocks were proposed to use an MLP on the channels for each pixel separately.

NiN blocks

- Recall that the inputs and outputs of convolutional layers consist of ?-dimensional arrays

NiN blocks

- Recall that the inputs and outputs of convolutional layers consist of ?-dimensional arrays
  - with axes corresponding to the batch, channel, height, and width.

NiN blocks

- Recall that the inputs and outputs of convolutional layers consist of ?-dimensional arrays
  - with axes corresponding to the batch, channel, height, and width.
- Also recall that the inputs and outputs of fully-connected layers are typically ?-dimensional arrays

NiN blocks

- Recall that the inputs and outputs of convolutional layers consist of ?-dimensional arrays
  - with axes corresponding to the batch, channel, height, and width.
- Also recall that the inputs and outputs of fully-connected layers are typically ?-dimensional arrays
  - corresponding to the batch, and features.

NiN blocks

- Recall that the inputs and outputs of convolutional layers consist of ?-dimensional arrays
  - with axes corresponding to the batch, channel, height, and width.
- Also recall that the inputs and outputs of fully-connected layers are typically ?-dimensional arrays
  - corresponding to the batch, and features.
- The idea behind NiN is to apply a fully-connected layer at each pixel location (for each height and width)

NiN blocks

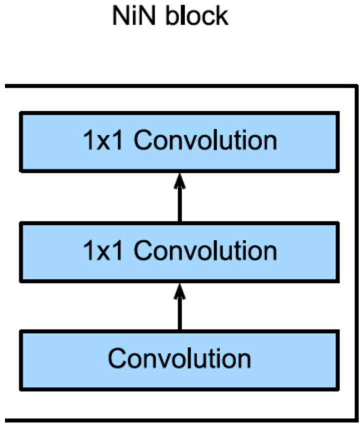
- Recall that the inputs and outputs of convolutional layers consist of ?-dimensional arrays
  - with axes corresponding to the batch, channel, height, and width.
- Also recall that the inputs and outputs of fully-connected layers are typically ?-dimensional arrays
  - corresponding to the batch, and features.
- The idea behind NiN is to apply a fully-connected layer at each pixel location (for each height and width)
- We could think of this as a 1x1 convolutional layer or as a fully-connected layer acting independently on each pixel location.

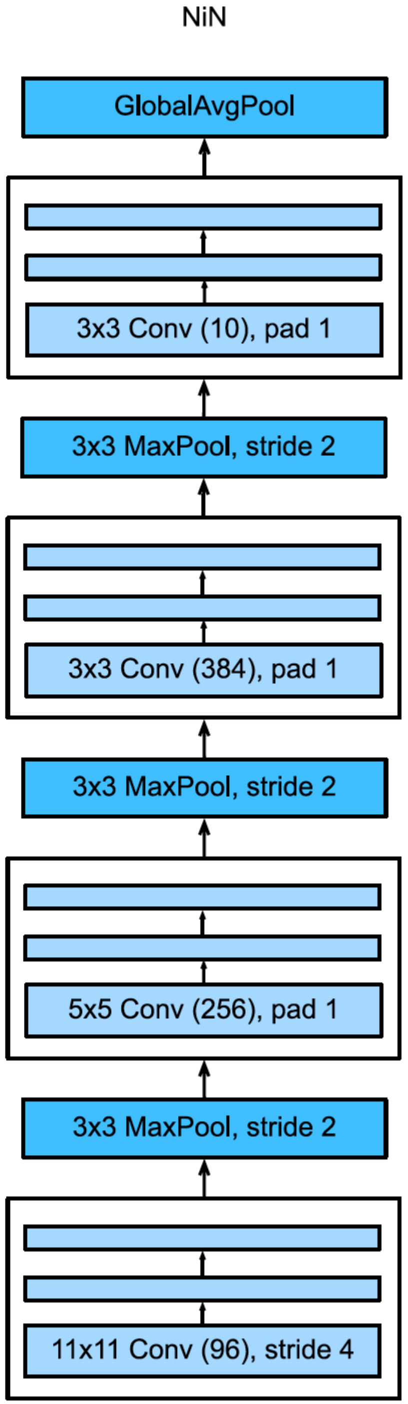
NiN blocks

- Recall that the inputs and outputs of convolutional layers consist of ?-dimensional arrays
  - with axes corresponding to the batch, channel, height, and width.
- Also recall that the inputs and outputs of fully-connected layers are typically ?-dimensional arrays
  - corresponding to the batch, and features.
- The idea behind NiN is to apply a fully-connected layer at each pixel location (for each height and width)
- We could think of this as a 1x1 convolutional layer or as a fully-connected layer acting independently on each pixel location.
- Another way to view this is to think of each element in the spatial dimension (height and width) as equivalent to an example and the channel as equivalent to a feature.

NiN models

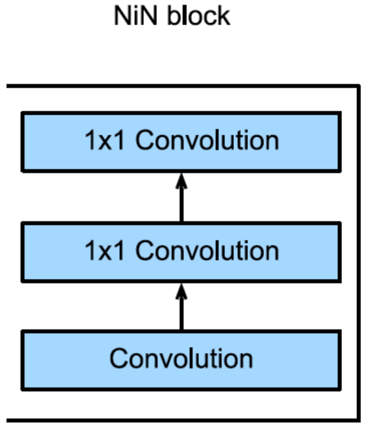
- The NiN block consists of one convolutional layer followed by two 1x1 convolutional layers that act as per-pixel fully-connected layers with ReLU activations. The convolution width of the first layer is typically set by the user. The subsequent widths are fixed to 1 x 1.

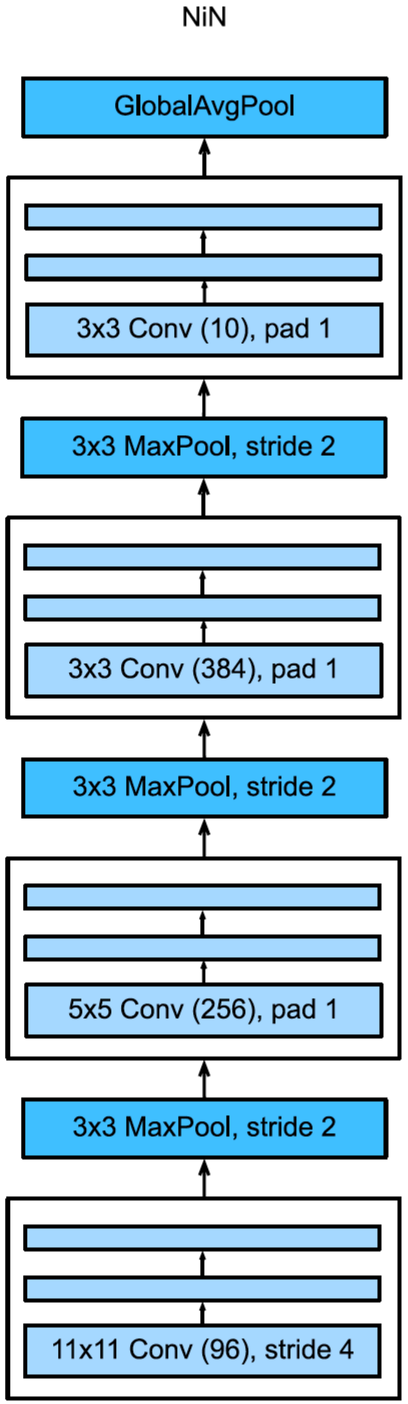




NiN models

- The NiN block consists of one convolutional layer followed by two 1x1 convolutional layers that act as per-pixel fully-connected layers with ReLU activations. The convolution width of the first layer is typically set by the user. The subsequent widths are fixed to 1 x 1.
- Once significant difference between NiN and AlexNet is that NiN avoids dense connections altogether. Instead, NiN uses an NiN block with a number of output channels equal to the number of label classes, followed by a global average pooling layer, yielding a vector of logits.





Networks with Parallel Concatenations (GoogleNet)

- In 2014, Szegedy et. al. (https://arxiv.org/pdf/1409.4842.pdf) won the ImageNet competition by proposing a structure that combined the strengths of the NiN and repeated blocks paradigms.

Networks with Parallel Concatenations (GoogleNet)

- In 2014, Szegedy et. al. (https://arxiv.org/pdf/1409.4842.pdf) won the ImageNet competition by proposing a structure that combined the strengths of the NiN and repeated blocks paradigms.
- One focus of the paper was to address the question of which sized convolutional kernels are best.

Networks with Parallel Concatenations (GoogleNet)

- In 2014, Szegedy et. al. (https://arxiv.org/pdf/1409.4842.pdf) wont the ImageNet competition by proposing a structure that combined the strengths of the NiN and repeated blocks paradigms.
- One focus of the paper was to address the question of which sized convolutional kernels are best.
- One insight in this paper was that sometimes it can be advantageous to employ a combination of variously-sized kernels.

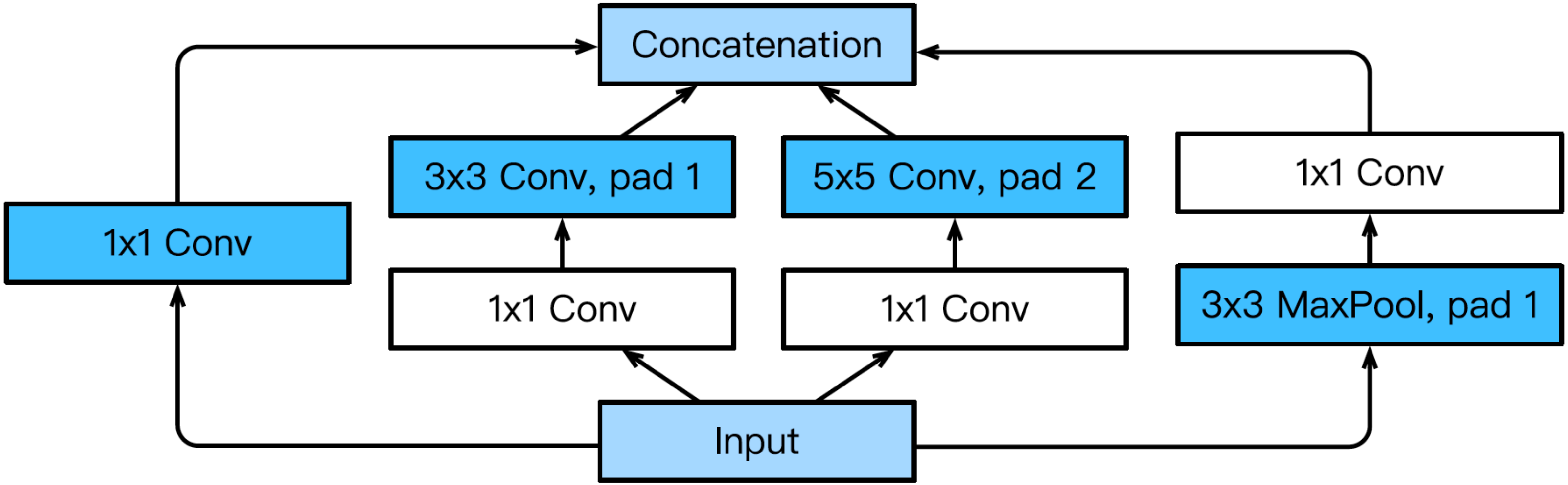
Inception Blocks

- The basic convolutional block in GoogLeNet is called an Inception block.

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

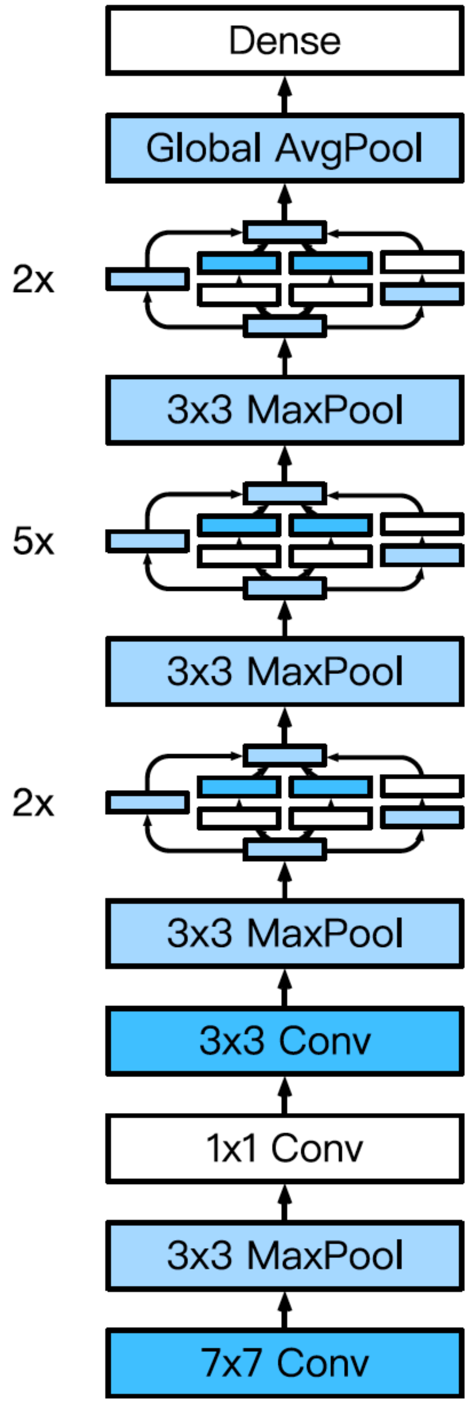
Inception Blocks

- The basic convolutional block in GoogLeNet is called an Inception block.



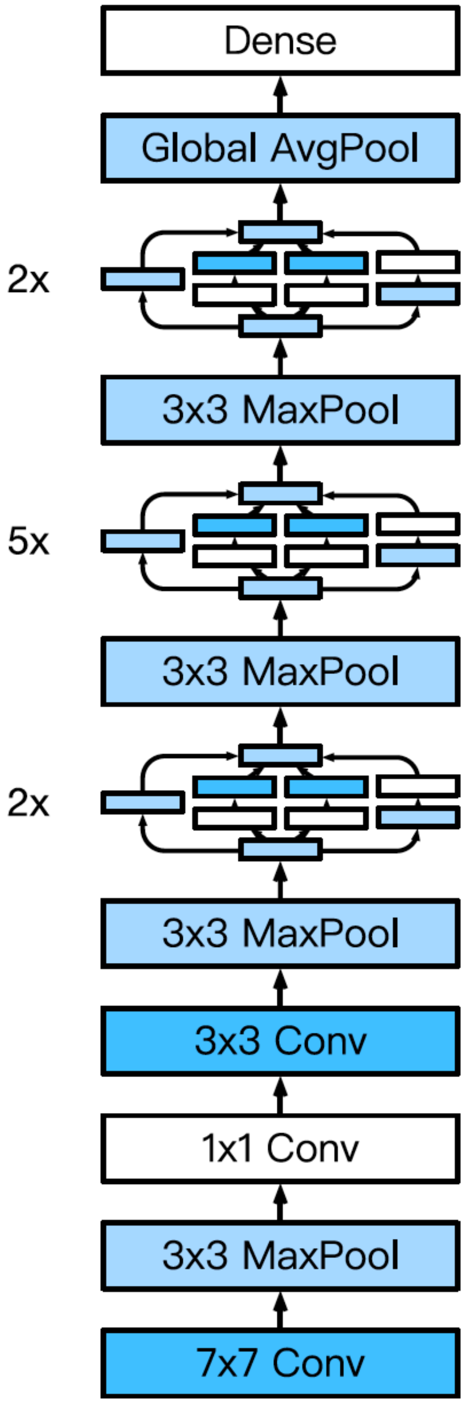
GoogleLeNet Model

- GoogLeNet uses a stack of a total of 9 inception blocks and global average pooling to generate its estimates.



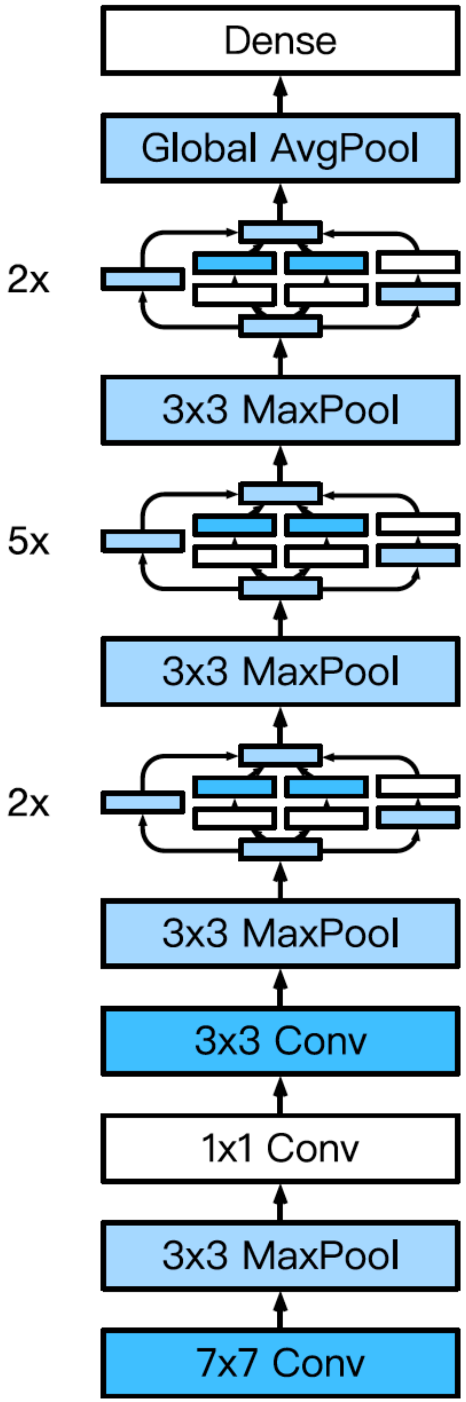
GoogleLeNet Model

- GoogLeNet uses a stack of a total of 9 inception blocks and global average pooling to generate its estimates.
- Maximum pooling between inception blocks reduced the dimensionality.



GoogleLeNet Model

- GoogLeNet uses a stack of a total of 9 inception blocks and global average pooling to generate its estimates.
- Maximum pooling between inception blocks reduced the dimensionality.
- The first part is identical to AlexNet and LeNet, the stack of blocks is inherited from VGG and the global average pooling avoids a stack of fully-connected layers at the end.



Batch Normalization

- Training deep models is difficult and getting them to converge in a reasonable amount of time can be tricky.

Batch Normalization

- Training deep models is difficult and getting them to converge in a reasonable amount of time can be tricky.
- Batch normalization is popular and effective technique that has been found to accelerate the convergence of deep nets.

Training Deep Networks

- Let’s review some of the practical challenges when training deep networks.

Training Deep Networks

- Let’s review some of the practical challenges when training deep networks.
- Data preprocessing often proves to be a crucial consideration for effective statistical modeling. Standardizing input data typically makes it easier to train models since parameters are a-priori at a similar scale.

Training Deep Networks

- Let’s review some of the practical challenges when training deep networks.
- Data preprocessing often proves to be a crucial consideration for effective statistical modeling. Standardizing input data typically makes it easier to train models since parameters are a-priori at a similar scale.
- For a typical MLP or CNN, as we train the model, the activations in intermediate layers of the network may assume different orders of magnitude. The authors of the batch normalization technique postulated that this drift in the distribution of activations could hamper the convergence of the network.

Training Deep Networks

- Let’s review some of the practical challenges when training deep networks.
- Data preprocessing often proves to be a crucial consideration for effective statistical modeling. Standardizing input data typically makes it easier to train models since parameters are a-priori at a similar scale.
- For a typical MLP or CNN, as we train the model, the activations in intermediate layers of the network may assume different orders of magnitude. The authors of the batch normalization technique postulated that this drift in the distribution of activations could hamper the convergence of the network.
- Deeper networks are complex and easily capable of overfitting. This means that regularization becomes more critical. Empirically, we note that even with dropout, models can overfit badly and we might benefit from other regularization heuristics.

Batch Normalization

- In 2015, a clever heuristic called batch normalization (BN) that has proved immensely useful for improving the reliability and speed of convergence when training deep models.

Batch Normalization

- In 2015, a clever heuristic called batch normalization (BN) that has proved immensely useful for improving the reliability and speed of convergence when training deep models.
- In each training iteration, BN normalizes the activations of each hidden layer node (on each layer where it is applied) by subtracting its mean and dividing by its standard deviation, estimating both based on the current minibatch.

Batch Normalization

- In 2015, a clever heuristic called batch normalization (BN) that has proved immensely useful for improving the reliability and speed of convergence when training deep models.
- In each training iteration, BN normalizes the activations of each hidden layer node (on each layer where it is applied) by subtracting its mean and dividing by its standard deviation, estimating both based on the current minibatch.
- In a nutshell, the idea in BN is to transform the activation at a given layer from $\mathbf{x}$ to

$$\mathrm{BN}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \hat{\mu}}{\hat{\sigma}} + \beta$$

Batch Normalization

- In 2015, a clever heuristic called batch normalization (BN) that has proved immensely useful for improving the reliability and speed of convergence when training deep models.
- In each training iteration, BN normalizes the activations of each hidden layer node (on each layer where it is applied) by subtracting its mean and dividing by its standard deviation, estimating both based on the current minibatch.
- In a nutshell, the idea in BN is to transform the activation at a given layer from $\mathbf{x}$ to

$$\mathrm{BN}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \hat{\mu}}{\hat{\sigma}} + \beta$$

- Consequently, the activations for intermediate layers cannot diverge any longer: we are actively rescaling them back to a given order of magnitude via μ and σ.

Batch Normalization

- Intuitively, it is hoped that this normalization allows us to be more aggressive in picking large learning rates.

Batch Normalization

- Intuitively, it is hoped that this normalization allows us to be more aggressive in picking large learning rates.
- To address the fact that in some cases the activations may actually need to differ from standardized data, BN also introduces scaling coefficients γ and offset β.

Batch Normalization

- Intuitively, it is hoped that this normalization allows us to be more aggressive in picking large learning rates.
- To address the fact that in some cases the activations may actually need to differ from standardized data, BN also introduces scaling coefficients γ and offset β.
- In principle, we might want to use all of our training data to estimate the mean and variance.

Batch Normalization

- Intuitively, it is hoped that this normalization allows us to be more aggressive in picking large learning rates.
- To address the fact that in some cases the activations may actually need to differ from standardized data, BN also introduces scaling coefficients γ and offset β.
- In principle, we might want to use all of our training data to estimate the mean and variance.
- However, the activations corresponding to each example change each time we update our model.

Batch Normalization

- Intuitively, it is hoped that this normalization allows us to be more aggressive in picking large learning rates.
- To address the fact that in some cases the activations may actually need to differ from standardized data, BN also introduces scaling coefficients γ and offset β.
- In principle, we might want to use all of our training data to estimate the mean and variance.
- However, the activations corresponding to each example change each time we update our model.
- To remedy this problem BN uses only the current minibatch for estimating mean and variance.

$$\hat{\mu}_{\mathcal{B}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{\mathbf{x} \in \mathcal{B}} \mathbf{x} \text{ and } \hat{\sigma}_{\mathcal{B}}^2 \leftarrow \frac{1}{|\mathcal{B}|} \sum_{\mathbf{x} \in \mathcal{B}} (\mathbf{x} - \mu_{\mathcal{B}})^2 + \epsilon$$

Batch Normalization Layers

- The batch normalization methods for fully-connected layers and convolutional layers are slightly different.

Batch Normalization Layers

- The batch normalization methods for fully-connected layers and convolutional layers are slightly different.
- This is due to the dimensionality of the data generated by convolutional layers

Batch Normalization Layers

- The batch normalization methods for fully-connected layers and convolutional layers are slightly different.
- This is due to the dimensionality of the data generated by convolutional layers
- **Fully-connected Layers:**
  - Usually we apply the batch normalization layer between the affine transformation and the activation function in a fully-connected layer.

Batch Normalization Layers

- The batch normalization methods for fully-connected layers and convolutional layers are slightly different.
- This is due to the dimensionality of the data generated by convolutional layers
- **Fully-connected Layers:**
  - Usually we apply the batch normalization layer between the affine transformation and the activation function in a fully-connected layer.
  - This yields the following variant of BN:

$$\mathbf{y} = \phi(\mathrm{BN}(\mathbf{x})) = \phi(\mathrm{BN}(\mathbf{W}\mathbf{u} + \mathbf{b}))$$

Batch Normalization Layers

- **Convolutional Layers:**

Batch Normalization Layers

- **Convolutional Layers:**
  - BN occurs after the convolution computation and before the application of the activation function.

Batch Normalization Layers

- **Convolutional Layers:**
  - BN occurs after the convolution computation and before the application of the activation function.
  - If the convolution computation outputs multiple channels, we need to carry out batch normalization for each of the outputs of these channels, and each channel has an independent scale parameter and shift parameter, both of which are scalars.

Batch Normalization Layers

- **Convolutional Layers:**
  - BN occurs after the convolution computation and before the application of the activation function.
  - If the convolution computation outputs multiple channels, we need to carry out batch normalization for each of the outputs of these channels, and each channel has an independent scale parameter and shift parameter, both of which are scalars.
  - Assume that there are m examples in the mini-batch.

Batch Normalization Layers

- **Convolutional Layers:**
  - BN occurs after the convolution computation and before the application of the activation function.
  - If the convolution computation outputs multiple channels, we need to carry out batch normalization for each of the outputs of these channels, and each channel has an independent scale parameter and shift parameter, both of which are scalars.
  - Assume that there are m examples in the mini-batch.
  - On a single channel, we assume that the height and width of the convolution computation output are p and q, respectively.

Batch Normalization Layers

- **Convolutional Layers:**
  - BN occurs after the convolution computation and before the application of the activation function.
  - If the convolution computation outputs multiple channels, we need to carry out batch normalization for each of the outputs of these channels, and each channel has an independent scale parameter and shift parameter, both of which are scalars.
  - Assume that there are m examples in the mini-batch.
  - On a single channel, we assume that the height and width of the convolution computation output are p and q, respectively.
  - We need to carry out batch normalization for $m \times p \times q$ elements in this channel simultaneously.

Batch Normalization Layers

- **Convolutional Layers:**
  - BN occurs after the convolution computation and before the application of the activation function.
  - If the convolution computation outputs multiple channels, we need to carry out batch normalization for each of the outputs of these channels, and each channel has an independent scale parameter and shift parameter, both of which are scalars.
  - Assume that there are m examples in the mini-batch.
  - On a single channel, we assume that the height and width of the convolution computation output are p and q, respectively.
  - We need to carry out batch normalization for $m \times p \times q$ elements in this channel simultaneously.
  - While carrying out the standardization computation for these elements, we use the same mean and variance.

Batch Normalization During Prediction

- At prediction time, we might not have the luxury of computing offsets per batch—we might be required to make one prediction at a time.

Batch Normalization During Prediction

- At prediction time, we might not have the luxury of computing offsets per batch—we might be required to make one prediction at a time.
- Secondly, the uncertainty in μ and σ , as arising from a minibatch are

undesirable once we've trained the model.

Batch Normalization During Prediction

- At prediction time, we might not have the luxury of computing offsets per batch—we might be required to make one prediction at a time.
- Secondly, the uncertainty in μ and σ , as arising from a minibatch are

undesirable once we've trained the model.

- One way to mitigate this is to compute more stable estimates on

a larger set for once (e.g. via a moving average).

Batch Normalization During Prediction

- At prediction time, we might not have the luxury of computing offsets per batch—we might be required to make one prediction at a time.
- Secondly, the uncertainty in μ and σ , as arising from a minibatch are

undesirable once we've trained the model.

- One way to mitigate this is to compute more stable estimates on

a larger set for once (e.g. via a moving average).

- Consequently, BN behaves differently during training and at test time.

Residual Networks (ResNet)

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

Residual Networks (ResNet)

- Paper: https://arxiv.org/pdf/1512.03385.pdf
- Recent evidence reveals that network depth is of crucial importance.

Residual Networks (ResNet)

- Paper: https://arxiv.org/pdf/1512.03385.pdf
- Recent evidence reveals that network depth is of crucial importance.
- Driven by the significance of depth, a question arises: Is learning better networks as easy as stacking more layers?

Residual Networks (ResNet)

- Paper: https://arxiv.org/pdf/1512.03385.pdf
- Recent evidence reveals that network depth is of crucial importance.
- Driven by the significance of depth, a question arises: Is learning better networks as easy as stacking more layers?
- An obstacle to answering this question was the notorious problem of vanishing/exploding gradients

Residual Networks (ResNet)

- Paper: https://arxiv.org/pdf/1512.03385.pdf
- Recent evidence reveals that network depth is of crucial importance.
- Driven by the significance of depth, a question arises: Is learning better networks as easy as stacking more layers?
- An obstacle to answering this question was the notorious problem of vanishing/exploding gradients
- When deeper networks are able to start converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly

Residual Networks (ResNet)

- Paper: https://arxiv.org/pdf/1512.03385.pdf
- Recent evidence reveals that network depth is of crucial importance.
- Driven by the significance of depth, a question arises: Is learning better networks as easy as stacking more layers?
- An obstacle to answering this question was the notorious problem of vanishing/exploding gradients
- When deeper networks are able to start converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly

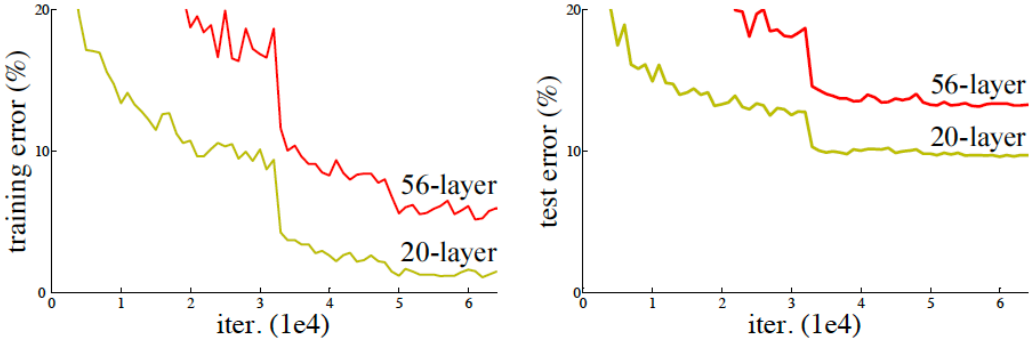


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena

Motivation Behind ResNet

- The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize.

Motivation Behind ResNet

- The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize.
- Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it.

Motivation Behind ResNet

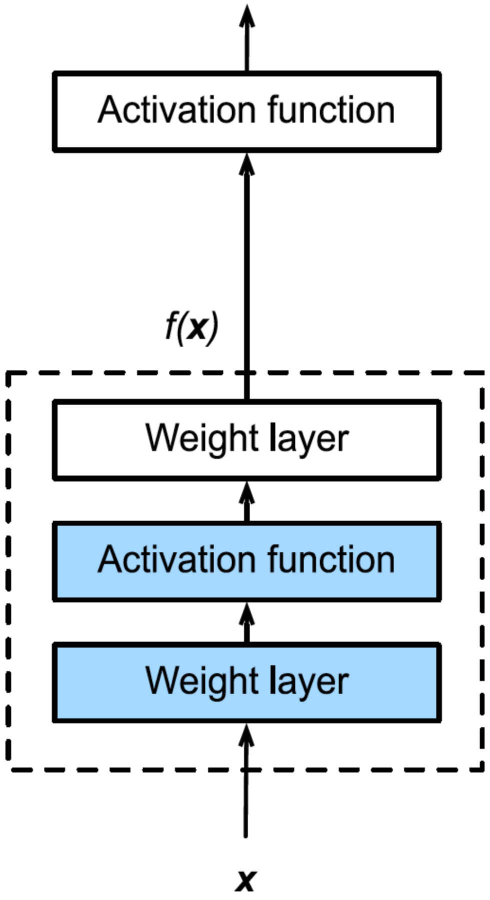
- The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize.
- Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it.
- There exists a solution by construction to the deeper model: the added layers are identity mapping, and the other layers are copied from the learned shallower model.

Motivation Behind ResNet

- The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize.
- Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it.
- There exists a solution by construction to the deeper model: the added layers are identity mapping, and the other layers are copied from the learned shallower model.
- The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart.

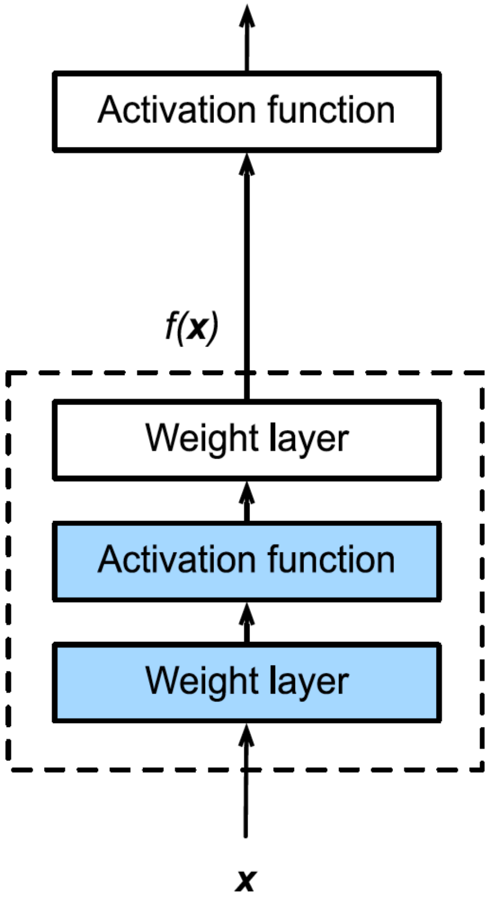
Residual Blocks

- Let us focus on a local neural network.



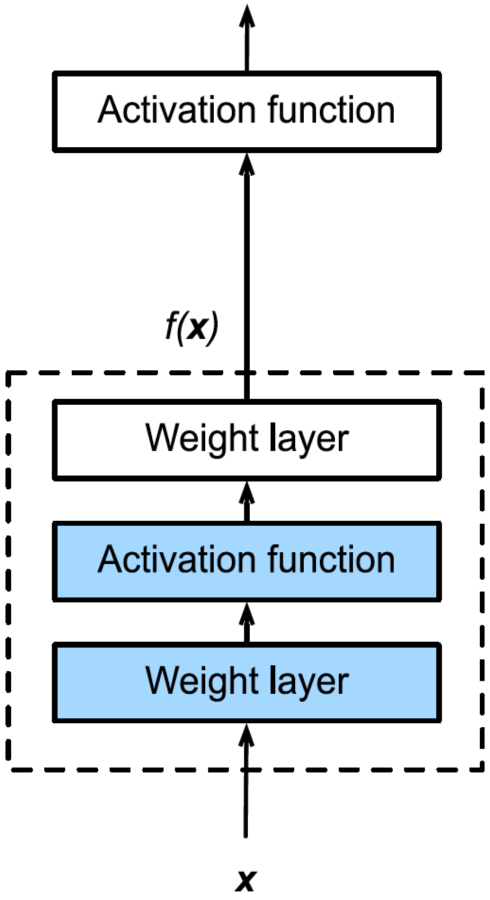
Residual Blocks

- Let us focus on a local neural network.
- Assume that the ideal mapping we want to obtain by learning is $f(\boldsymbol{x})$.



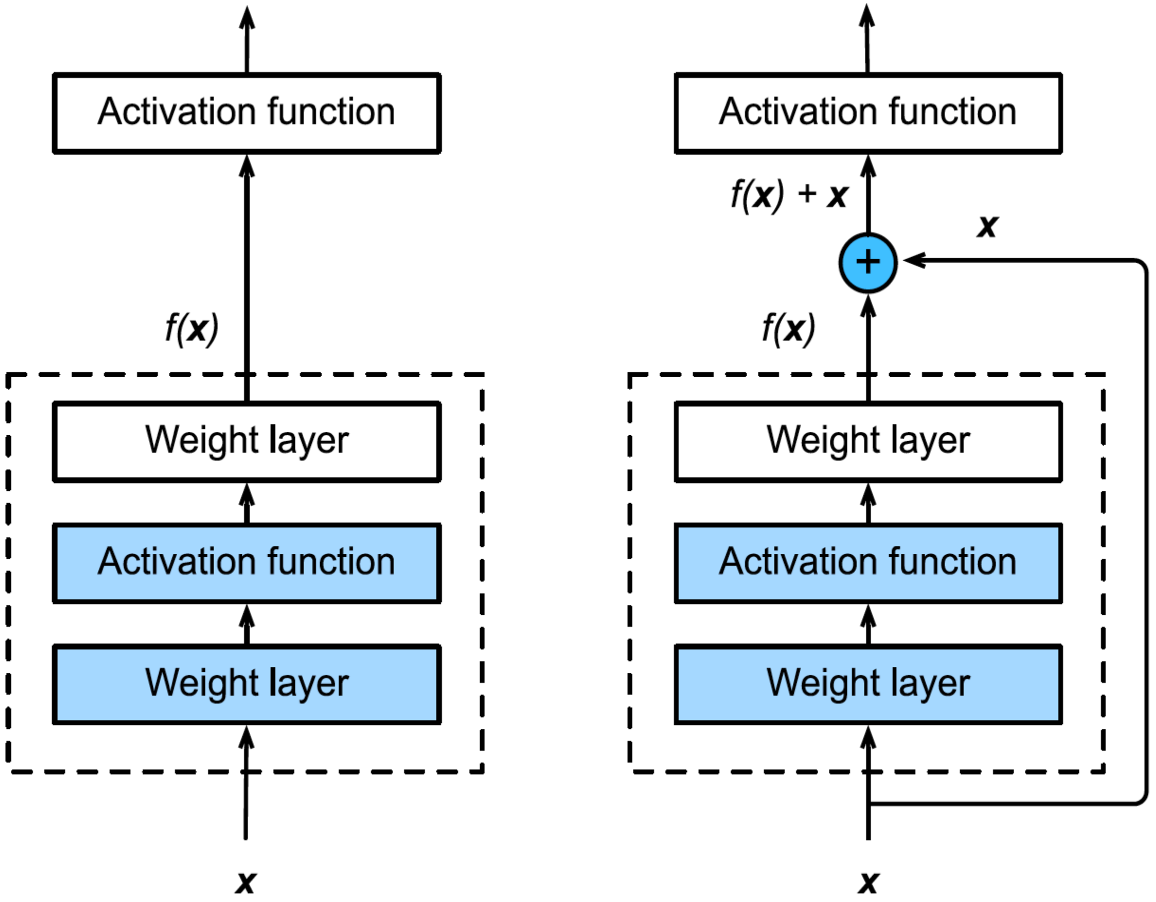
Residual Blocks

- Let us focus on a local neural network.
- Assume that the ideal mapping we want to obtain by learning is $f(\boldsymbol{x})$.
- Fitting this can be tricky if we do not need it and we would much rather retain the input $\boldsymbol{x}$.



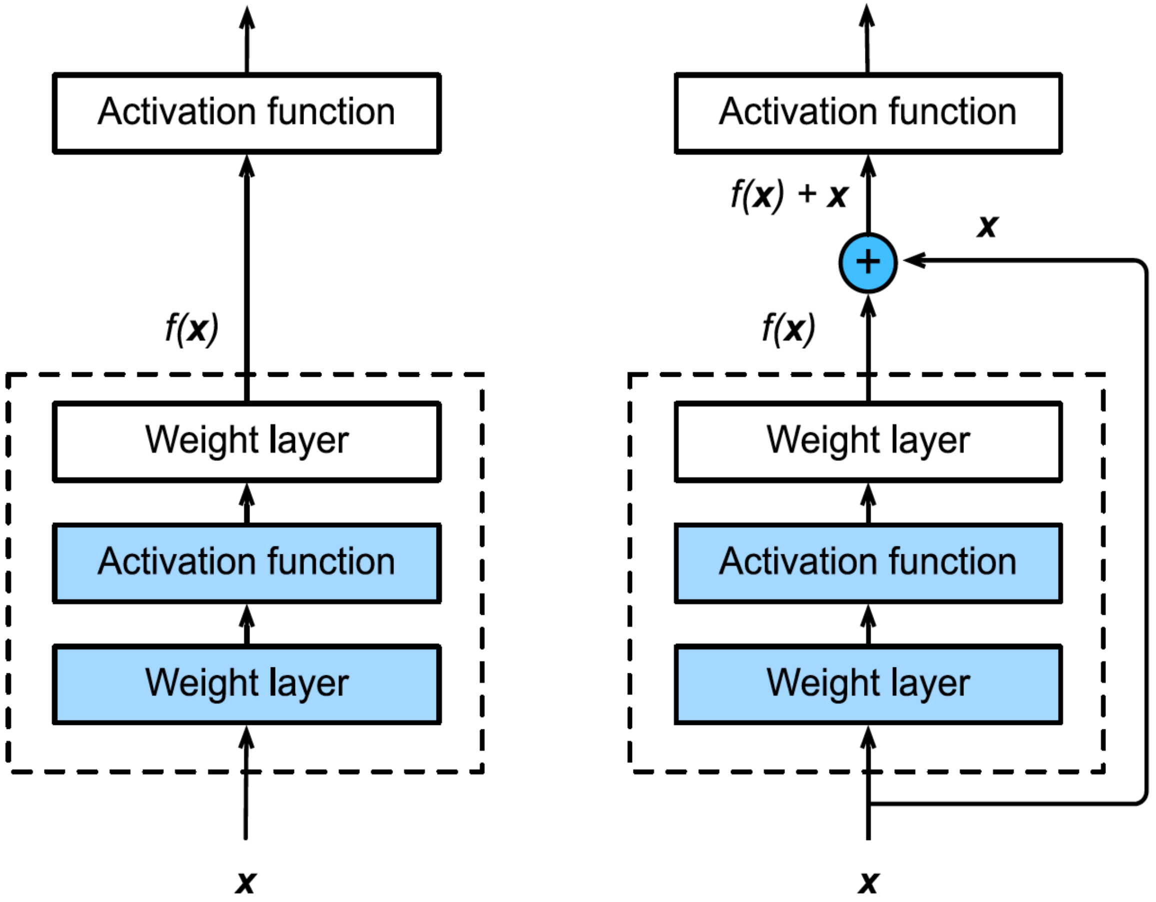
Residual Blocks

- Let us focus on a local neural network.
- Assume that the ideal mapping we want to obtain by learning is $f(\boldsymbol{x})$.
- Fitting this can be tricky if we do not need it and we would much rather retain the input $\boldsymbol{x}$.
- The portion within the dotted-line box in the right image now only needs to parametrize the deviation from the identity



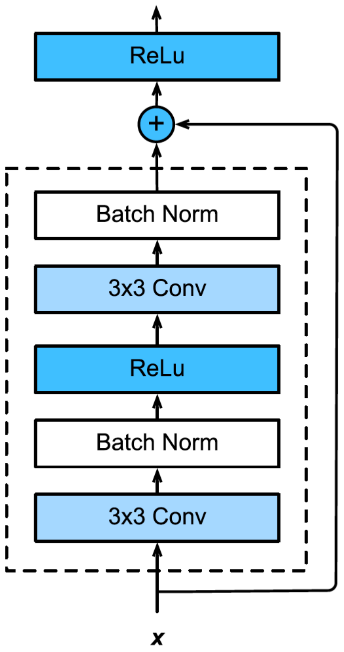
Residual Blocks

- Let us focus on a local neural network.
- Assume that the ideal mapping we want to obtain by learning is $f(\boldsymbol{x})$.
- Fitting this can be tricky if we do not need it and we would much rather retain the input $\boldsymbol{x}$.
- The portion within the dotted-line box in the right image now only needs to parametrize the deviation from the identity
- In practice, the residual mapping is often easier to optimize.



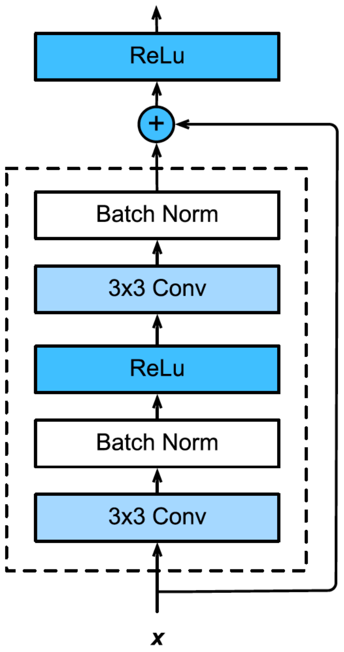
Residual Blocks

- ResNet follows VGG’s full 3 x 3 convolutional layer design.



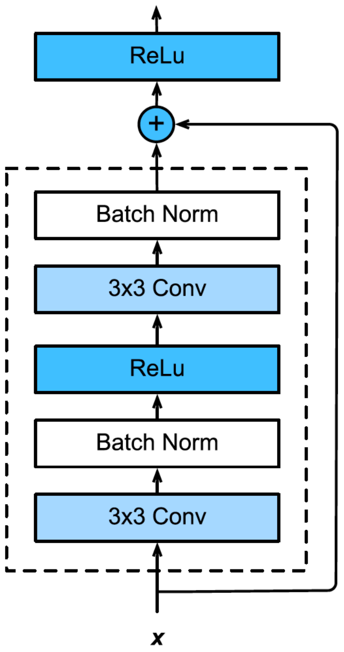
Residual Blocks

- ResNet follows VGG's full 3 x 3 convolutional layer design.
- The residual block has two 3 x 3 convolutional layers with the same number of output channels.



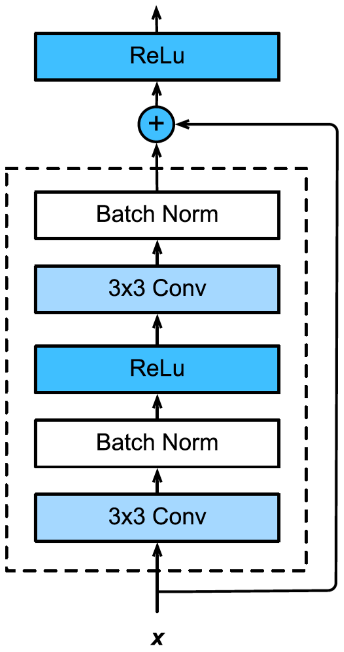
Residual Blocks

- ResNet follows VGG's full 3 x 3 convolutional layer design.
- The residual block has two 3 x 3 convolutional layers with the same number of output channels.
- Each convolutional layer is followed by a batch normalization layer and a ReLU activation function.



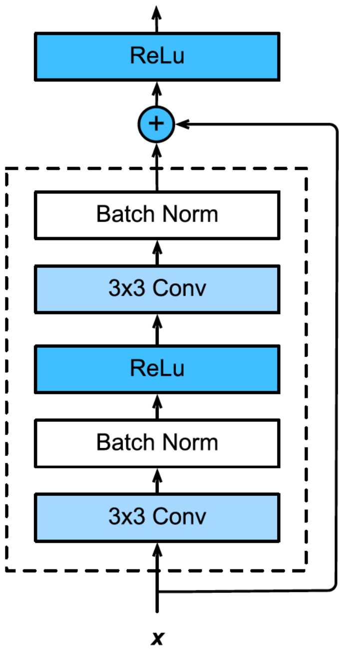
Residual Blocks

- ResNet follows VGG's full 3 x 3 convolutional layer design.
- The residual block has two 3 x 3 convolutional layers with the same number of output channels.
- Each convolutional layer is followed by a batch normalization layer and a ReLU activation function.
- Then, we skip these two convolution operations and add the input directly before the final ReLU activation function.



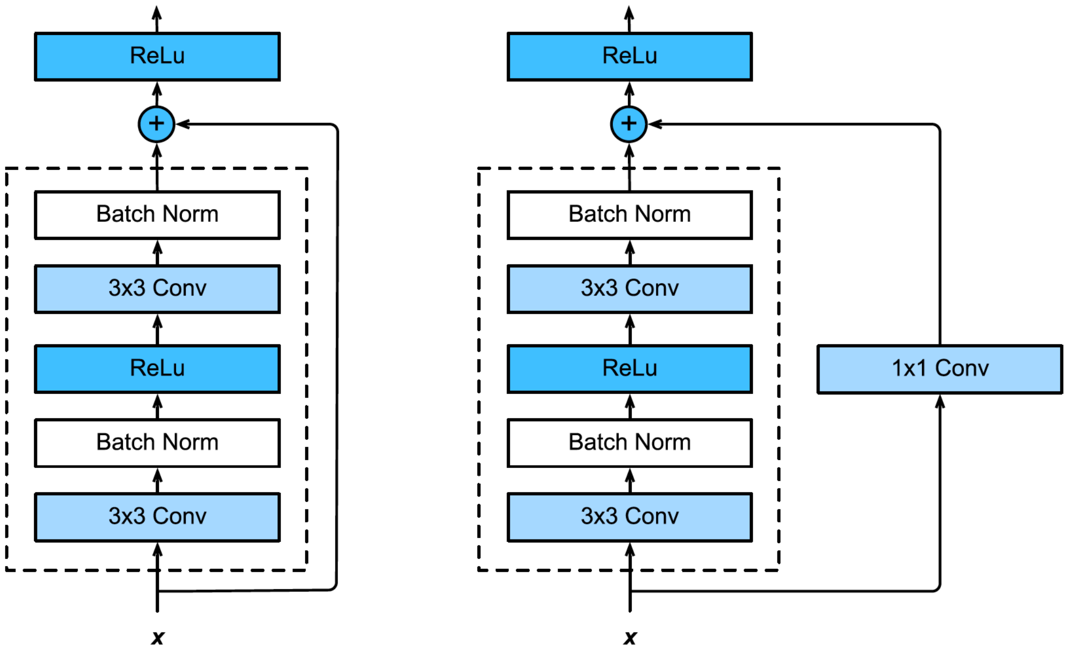
Residual Blocks

- ResNet follows VGG’s full 3 x 3 convolutional layer design.
- The residual block has two 3 x 3 convolutional layers with the same number of output channels.
- Each convolutional layer is followed by a batch normalization layer and a ReLU activation function.
- Then, we skip these two convolution operations and add the input directly before the final ReLU activation function.
- This kind of design requires that the output of the two convolutional layers be of the same shape as the input, so that they can be added together.



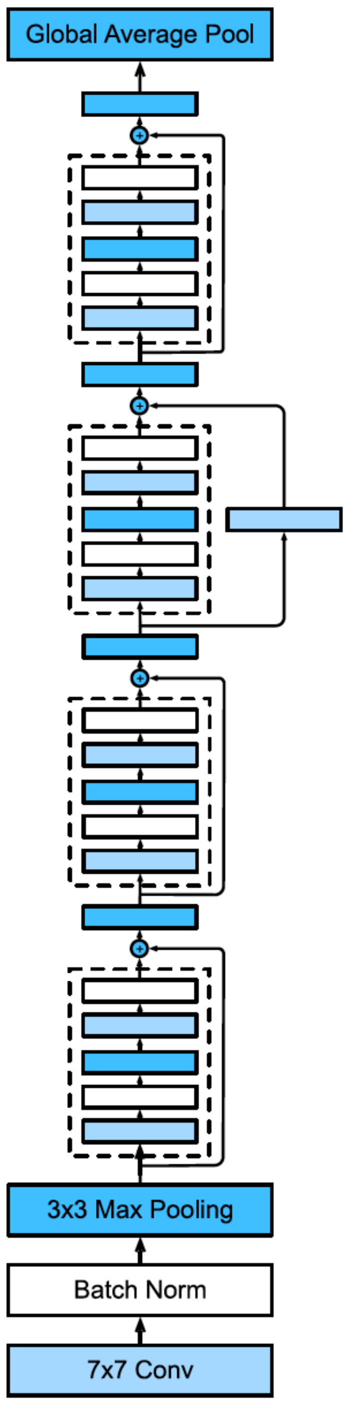
Residual Blocks

- ResNet follows VGG's full 3 x 3 convolutional layer design.
- The residual block has two 3 x 3 convolutional layers with the same number of output channels.
- Each convolutional layer is followed by a batch normalization layer and a ReLU activation function.
- Then, we skip these two convolution operations and add the input directly before the final ReLU activation function.
- This kind of design requires that the output of the two convolutional layers be of the same shape as the input, so that they can be added together.
- If we want to change the number of channels or the stride, we need to introduce an additional 1x1 convolutional layer to transform the input into the desired shape for the addition operation.



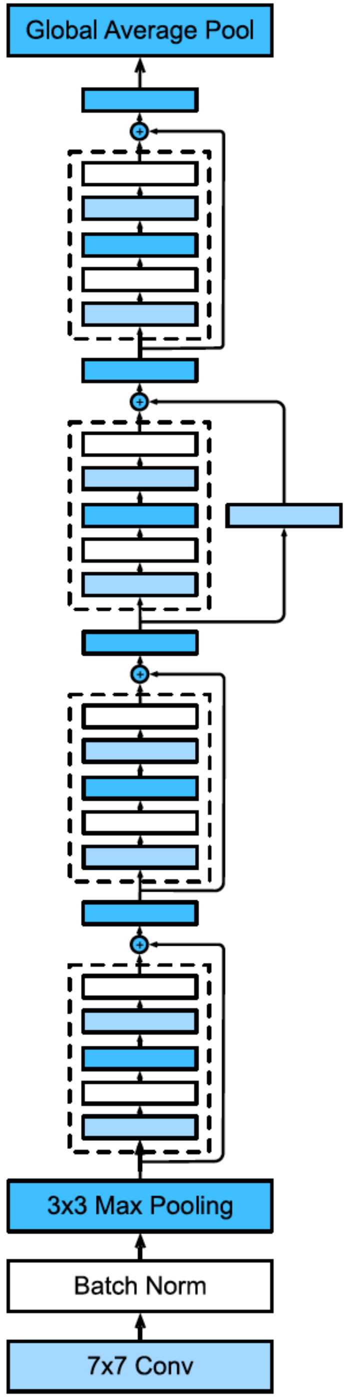
ResNet Model

- The first two layers of ResNet are the same as those of the GoogLeNet



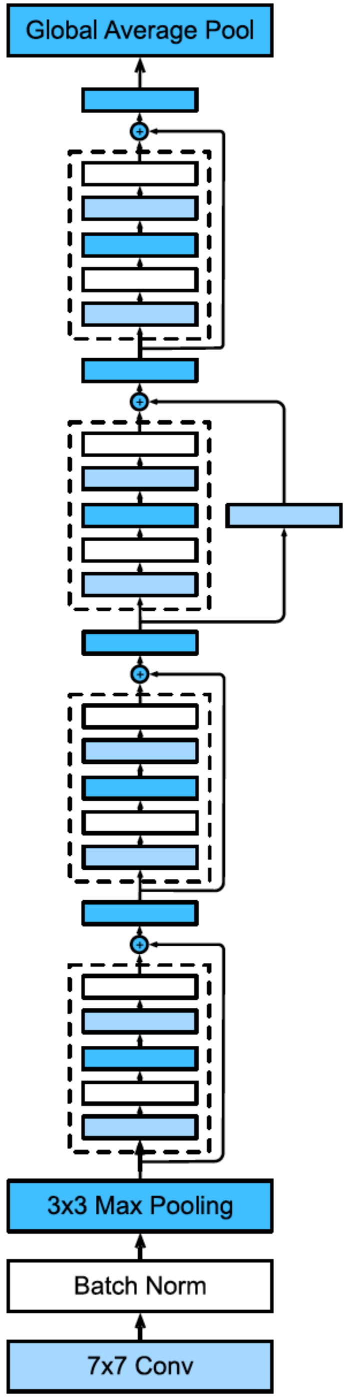
ResNet Model

- The first two layers of ResNet are the same as those of the GoogLeNet
- The difference is the batch normalization layer added after each convolutional layer in ResNet.



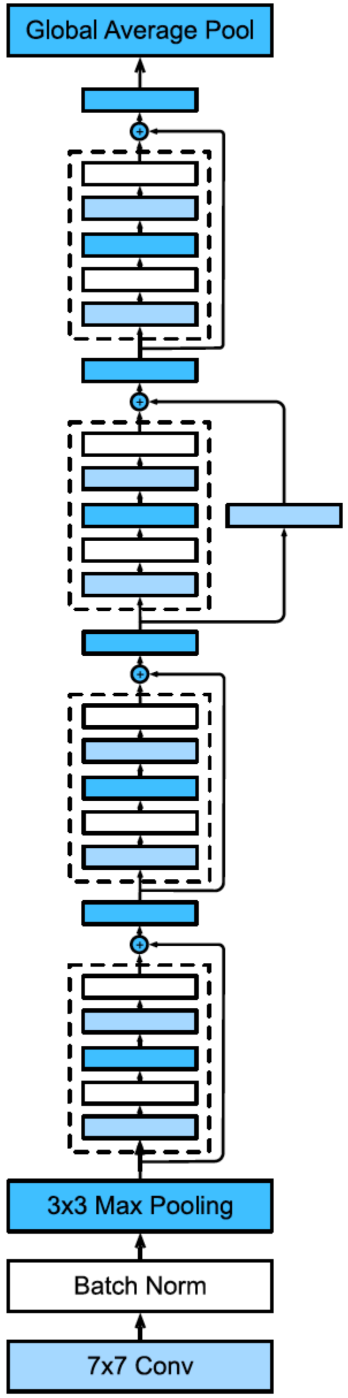
ResNet Model

- The first two layers of ResNet are the same as those of the GoogLeNet
- The difference is the batch normalization layer added after each convolutional layer in ResNet.
- ResNet uses four modules made up of residual blocks, each of which uses several residual blocks with the same number of output channels.

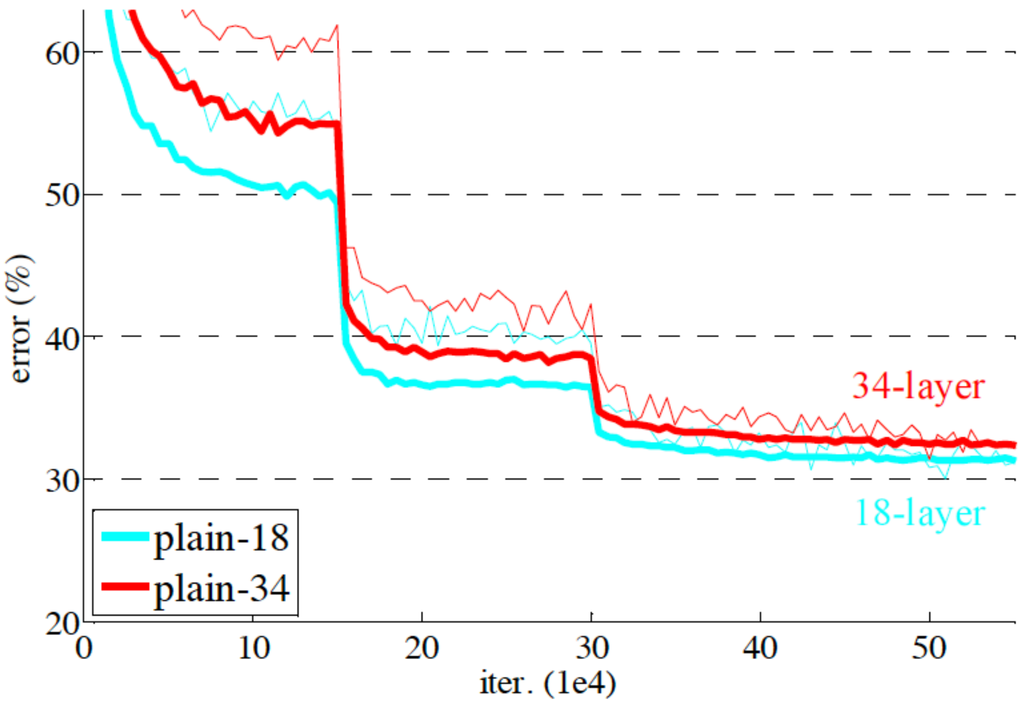


ResNet Model

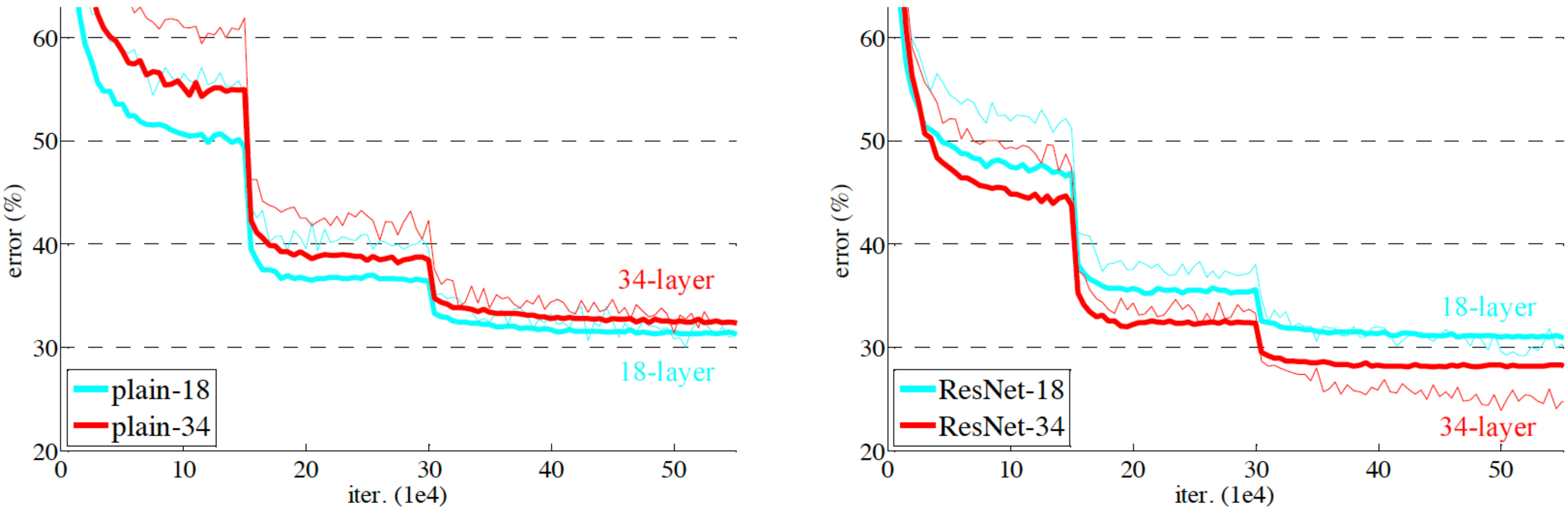
- The first two layers of ResNet are the same as those of the GoogLeNet
- The difference is the batch normalization layer added after each convolutional layer in ResNet.
- ResNet uses four modules made up of residual blocks, each of which uses several residual blocks with the same number of output channels.
- There are 4 convolutional layers in each module (excluding the 1 1 convolutional layer). Together with the first convolutional layer and the final fully connected layer, there are 18 layers in total.



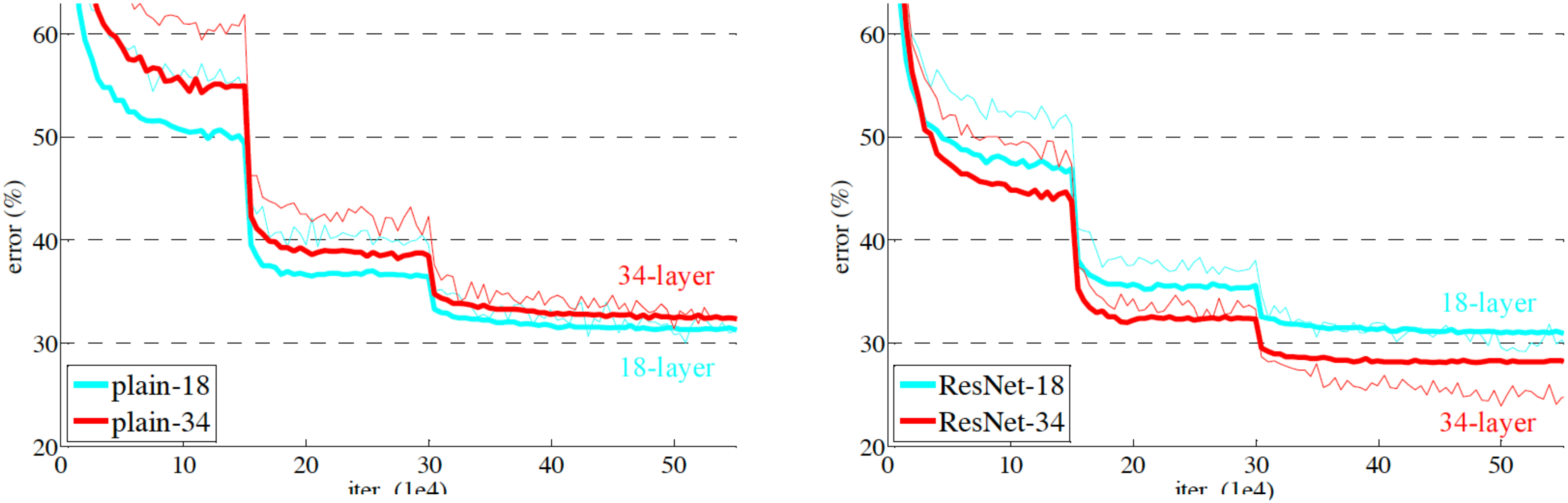
Results Using ResNet



Results Using ResNet



Results Using ResNet



	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Table 2. Top-1 error (%, 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

Densely Connected Networks (DenseNet)

- ResNet significantly changed the view of how to parametrize the functions in deep networks.

Densely Connected Networks (DenseNet)

- ResNet significantly changed the view of how to parametrize the functions in deep networks.
- DenseNet is to some extent the logical extension of this.

Densely Connected Networks (DenseNet)

- ResNet significantly changed the view of how to parametrize the functions in deep networks.
- DenseNet is to some extent the logical extension of this.
- Recall the Taylor expansion for functions. For scalars it can be written as

$$f(x) = f(0) + f'(x)x + \frac{1}{2}f''(x)x^2 + \frac{1}{6}f'''(x)x^3 + o(x^3)$$

Densely Connected Networks (DenseNet)

- ResNet significantly changed the view of how to parametrize the functions in deep networks.
- DenseNet is to some extent the logical extension of this.
- Recall the Taylor expansion for functions. For scalars it can be written as

$$f(x) = f(0) + f'(x)x + \frac{1}{2}f''(x)x^2 + \frac{1}{6}f'''(x)x^3 + o(x^3)$$

- In a similar vein, ResNet decomposes functions into $f(\mathbf{x}) = \mathbf{x} + g(\mathbf{x})$

Densely Connected Networks (DenseNet)

- ResNet significantly changed the view of how to parametrize the functions in deep networks.
- DenseNet is to some extent the logical extension of this.
- Recall the Taylor expansion for functions. For scalars it can be written as

$$f(x) = f(0) + f'(x)x + \frac{1}{2}f''(x)x^2 + \frac{1}{6}f'''(x)x^3 + o(x^3)$$

- In a similar vein, ResNet decomposes functions into $f(\mathbf{x}) = \mathbf{x} + g(\mathbf{x})$
- That is, ResNet decomposes f into a simple linear term and a more complex nonlinear one. What if we want to go beyond two terms?

Densely Connected Networks (DenseNet)

- ResNet significantly changed the view of how to parametrize the functions in deep networks.
- DenseNet is to some extent the logical extension of this.
- Recall the Taylor expansion for functions. For scalars it can be written as

$$f(x) = f(0) + f'(x)x + \frac{1}{2}f''(x)x^2 + \frac{1}{6}f'''(x)x^3 + o(x^3)$$

- In a similar vein, ResNet decomposes functions into $f(\mathbf{x}) = \mathbf{x} + g(\mathbf{x})$
- That is, ResNet decomposes f into a simple linear term and a more complex nonlinear one. What if we want to go beyond two terms?
- A solution was proposed by the paper: https://arxiv.org/pdf/1608.06993.pdf

Densely Connected Networks (DenseNet)

- The key difference between ResNet and DenseNet is that in the latter case outputs are concatenated rather than added.

$$\mathbf{x} \to [\mathbf{x}, f_1(\mathbf{x}), f_2(\mathbf{x}, f_1(\mathbf{x})), f_3(\mathbf{x}, f_1(\mathbf{x}), f_2(\mathbf{x}, f_1(\mathbf{x})), \ldots]$$

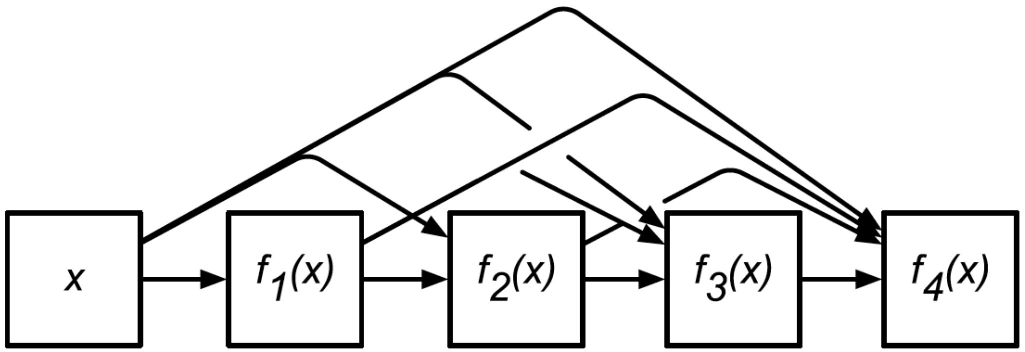


Fig. 9.7.2: Dense connections in DenseNet

Densely Connected Networks (DenseNet)

- The key difference between ResNet and DenseNet is that in the latter case outputs are concatenated rather than added.

$$\mathbf{x} \to [\mathbf{x}, f_1(\mathbf{x}), f_2(\mathbf{x}, f_1(\mathbf{x})), f_3(\mathbf{x}, f_1(\mathbf{x}), f_2(\mathbf{x}, f_1(\mathbf{x})), \ldots]$$

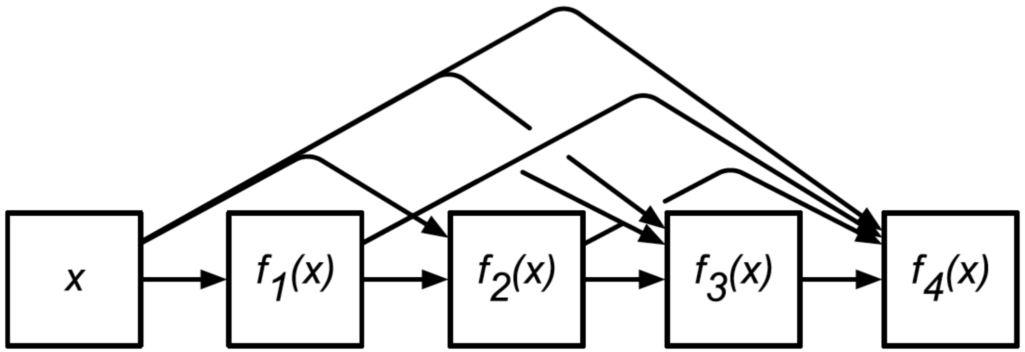


Fig. 9.7.2: Dense connections in DenseNet

- In the end, all these functions are combined in an MLP to reduce the number of features again.

Densely Connected Networks (DenseNet)

- The key difference between ResNet and DenseNet is that in the latter case outputs are concatenated rather than added.

$$\mathbf{x} \to [\mathbf{x}, f_1(\mathbf{x}), f_2(\mathbf{x}, f_1(\mathbf{x})), f_3(\mathbf{x}, f_1(\mathbf{x}), f_2(\mathbf{x}, f_1(\mathbf{x})), \ldots]$$

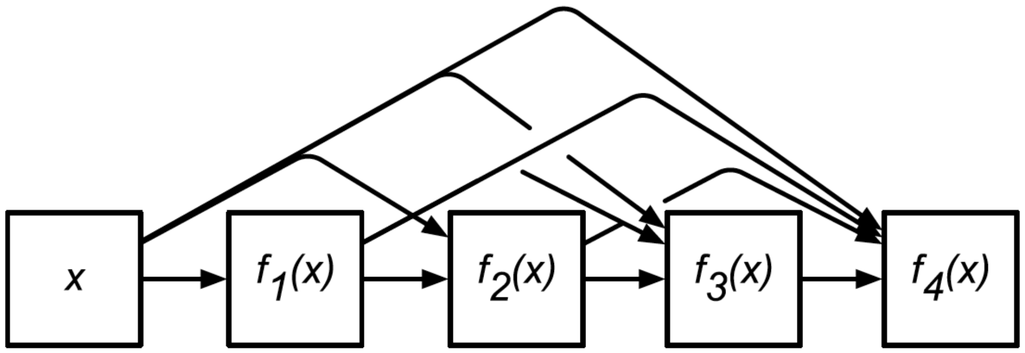


Fig. 9.7.2: Dense connections in DenseNet

- In the end, all these functions are combined in an MLP to reduce the number of features again.
- The main components that compose a DenseNet are dense blocks and transition layers. The former defines how the inputs and outputs are concatenated, while the latter controls the number of channels so that it is not too large.

DenseNet Model

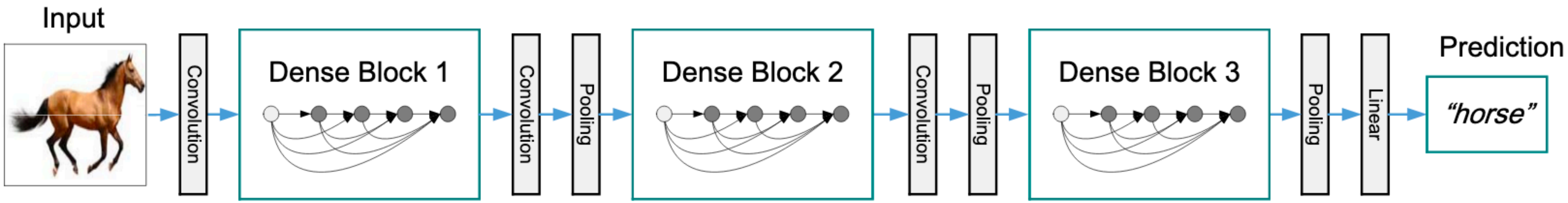


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.