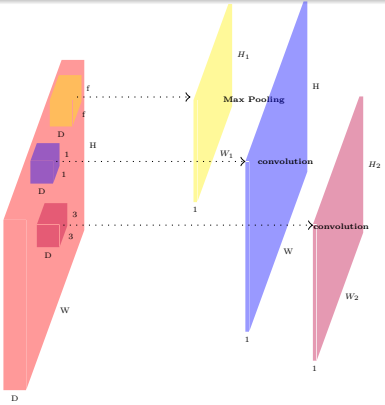
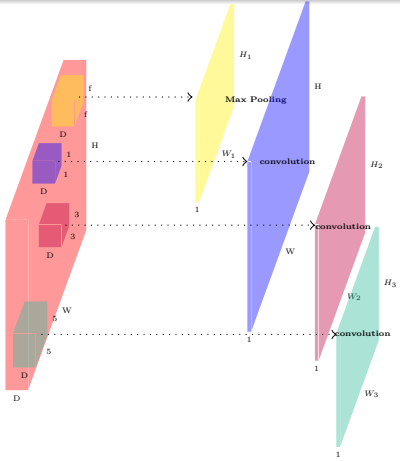
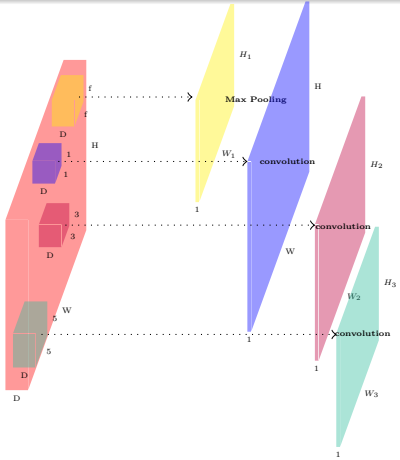


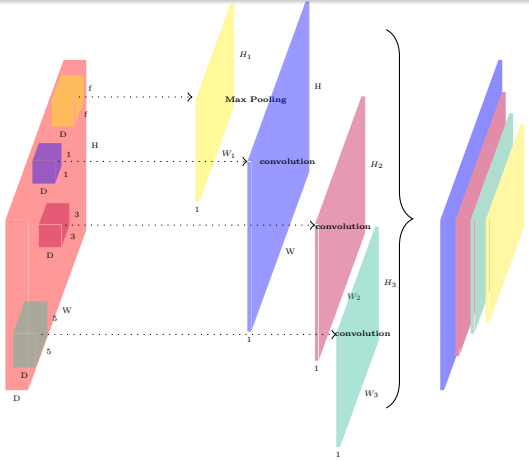
- Consider the output at a certain layer of a convolutional neural network
- After this layer we could apply a max-pooling layer
- Or a 1×1 convolution
- Or a 3×3 convolution



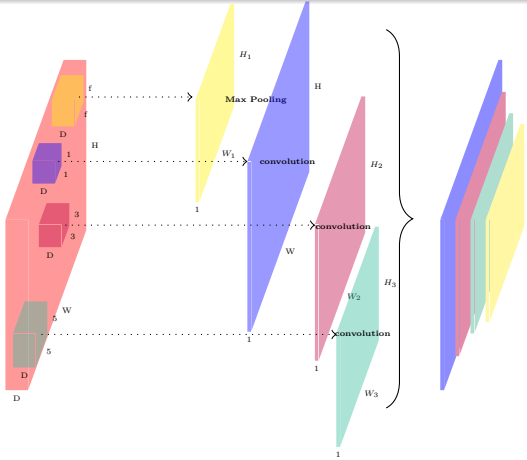
- Consider the output at a certain layer of a convolutional neural network
- After this layer we could apply a max-pooling layer
- Or a 1×1 convolution
- Or a 3×3 convolution
- Or a 5×5 convolution



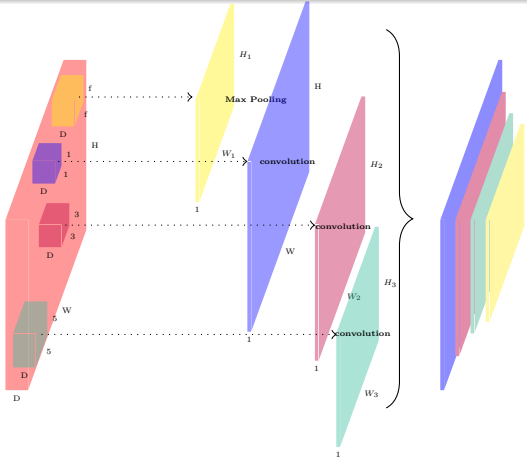
- Consider the output at a certain layer of a convolutional neural network
- After this layer we could apply a max-pooling layer
- Or a 1×1 convolution
- Or a 3×3 convolution
- Or a 5×5 convolution
- **Question:** Why choose between these options (convolution, maxpooling, filter sizes)?



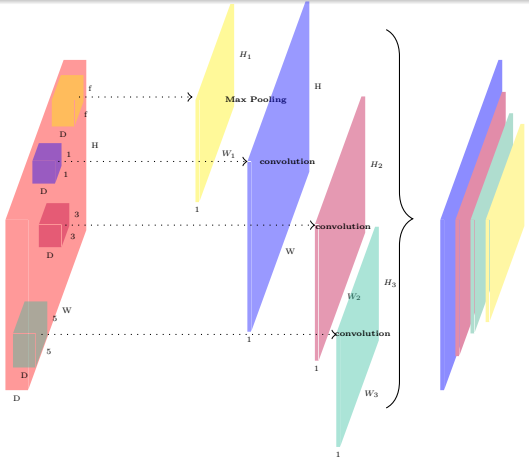
- Consider the output at a certain layer of a convolutional neural network
- After this layer we could apply a max-pooling layer
- Or a 1×1 convolution
- Or a 3×3 convolution
- Or a 5×5 convolution
- **Question:** Why choose between these options (convolution, maxpooling, filter sizes)?
- **Idea:** Why not apply all of them at the same time and then concatenate the feature maps?



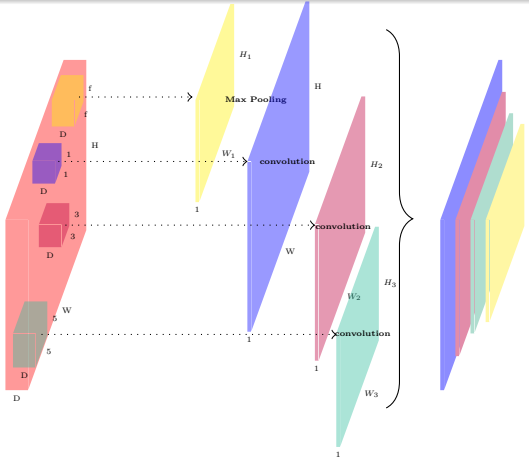
- Well this naive idea could result in a large number of computations



- Well this naive idea could result in a large number of computations
- If $P = 0$ & $S = 1$ then convolving a $W \times H \times D$ input with a $F \times F \times D$ filter results in a $(W - F + 1)(H - F + 1)$ sized output

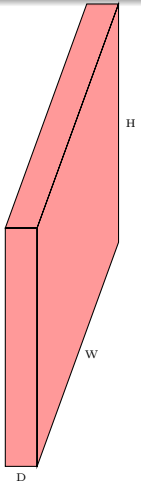


- Well this naive idea could result in a large number of computations
- If $P = 0$ & $S = 1$ then convolving a $W \times H \times D$ input with a $F \times F \times D$ filter results in a $(W - F + 1)(H - F + 1)$ sized output
- Each element of the output requires $O(F \times F \times D)$ computations

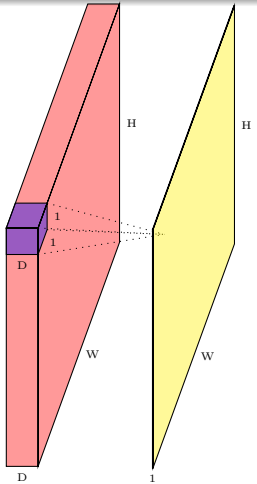


- Well this naive idea could result in a large number of computations
- If $P = 0$ & $S = 1$ then convolving a $W \times H \times D$ input with a $F \times F \times D$ filter results in a $(W - F + 1)(H - F + 1)$ sized output
- Each element of the output requires $O(F \times F \times D)$ computations
- Can we reduce the number of computations?

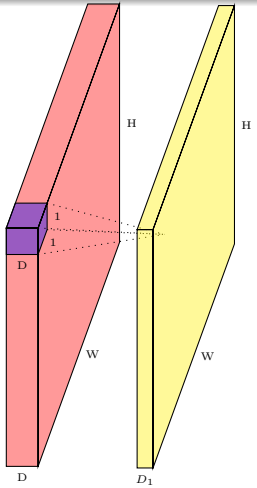
- Yes, by using 1×1 convolutions



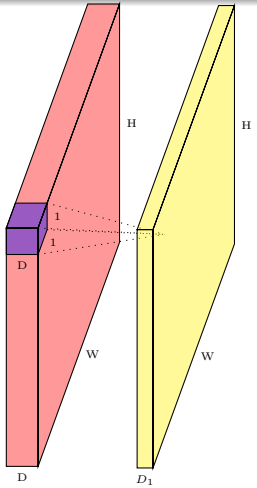
- Yes, by using 1×1 convolutions
- Huh?? What does a 1×1 convolution do ?



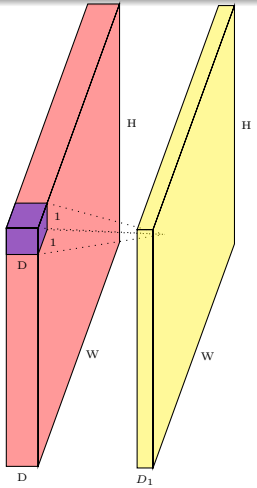
- Yes, by using 1×1 convolutions
- Huh?? What does a 1×1 convolution do ?
- It aggregates along the depth



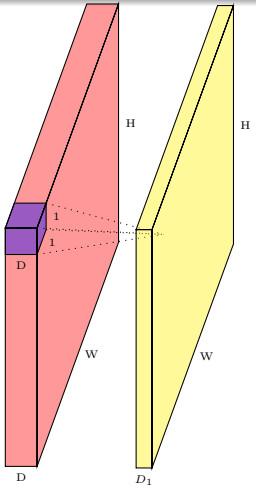
- Yes, by using 1×1 convolutions
- Huh?? What does a 1×1 convolution do ?
- It aggregates along the depth
- So convolving a $D \times W \times H$ input with D_1 1×1 ($D_1 < D$) filters will result in a $D_1 \times W \times H$ output ($S = 1, P = 0$)



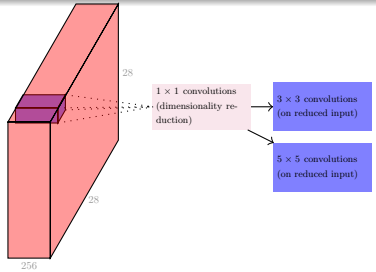
- Yes, by using 1×1 convolutions
- Huh?? What does a 1×1 convolution do ?
- It aggregates along the depth
- So convolving a $D \times W \times H$ input with D_1 1×1 ($D_1 < D$) filters will result in a $D_1 \times W \times H$ output ($S = 1, P = 0$)
- If $D_1 < D$ then this effectively reduces the dimension of the input and hence the computations



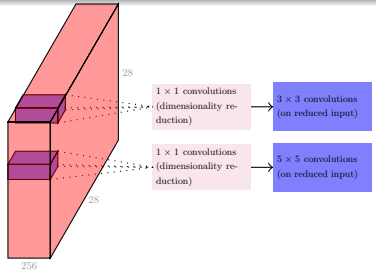
- Yes, by using 1×1 convolutions
- Huh?? What does a 1×1 convolution do ?
- It aggregates along the depth
- So convolving a $D \times W \times H$ input with D_1 1×1 ($D_1 < D$) filters will result in a $D_1 \times W \times H$ output ($S = 1, P = 0$)
- If $D_1 < D$ then this effectively reduces the dimension of the input and hence the computations
- Specifically instead of $O(F \times F \times D)$ we will need $O(F \times F \times D_1)$ computations



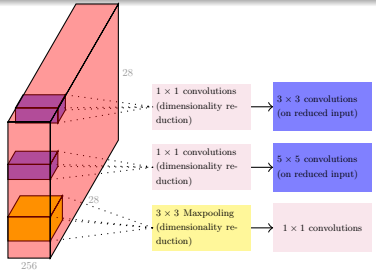
- Yes, by using 1×1 convolutions
- Huh?? What does a 1×1 convolution do ?
- It aggregates along the depth
- So convolving a $D \times W \times H$ input with D_1 1×1 ($D_1 < D$) filters will result in a $D_1 \times W \times H$ output ($S = 1, P = 0$)
- If $D_1 < D$ then this effectively reduces the dimension of the input and hence the computations
- Specifically instead of $O(F \times F \times D)$ we will need $O(F \times F \times D_1)$ computations
- We could then apply subsequent 3×3, 5×5 filter on this reduced output



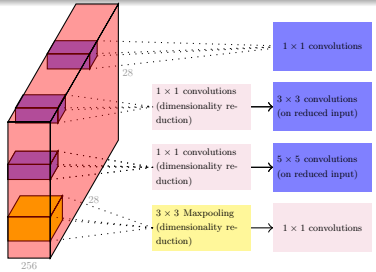
- But we might want to use different dimensionality reductions before the 3×3 and 5×5 filters



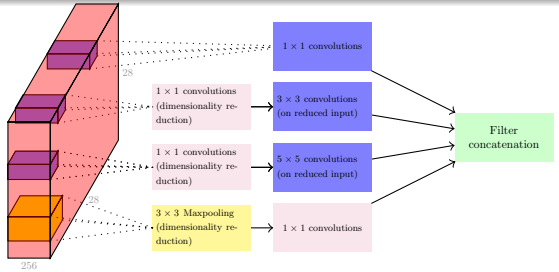
- But we might want to use different dimensionality reductions before the 3×3 and 5×5 filters
- So we can use D_1 and D_2 1×1 filters before the 3×3 and 5×5 filters respectively



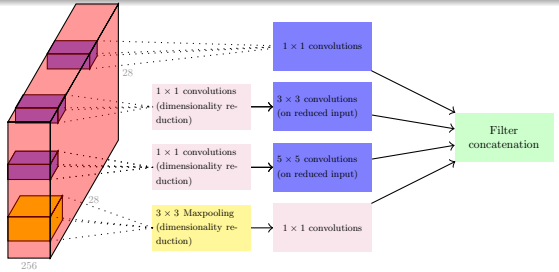
- But we might want to use different dimensionality reductions before the 3×3 and 5×5 filters
- So we can use D_1 and D_2 1×1 filters before the 3×3 and 5×5 filters respectively
- We can then add the maxpooling layer followed by dimensionality reduction



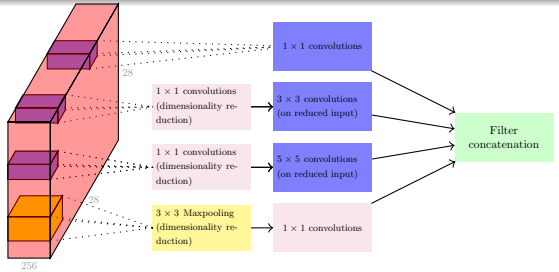
- But we might want to use different dimensionality reductions before the 3×3 and 5×5 filters
- So we can use D_1 and D_2 1×1 filters before the 3×3 and 5×5 filters respectively
- We can then add the maxpooling layer followed by dimensionality reduction
- And a new set of 1×1 convolutions



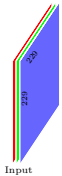
- But we might want to use different dimensionality reductions before the 3×3 and 5×5 filters
- So we can use D_1 and D_2 1×1 filters before the 3×3 and 5×5 filters respectively
- We can then add the maxpooling layer followed by dimensionality reduction
- And a new set of 1×1 convolutions
- And finally we concatenate all these layers

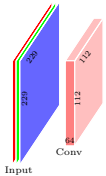


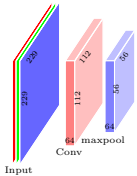
- But we might want to use different dimensionality reductions before the 3×3 and 5×5 filters
- So we can use D_1 and D_2 1×1 filters before the 3×3 and 5×5 filters respectively
- We can then add the maxpooling layer followed by dimensionality reduction
- And a new set of 1×1 convolutions
- And finally we concatenate all these layers
- This is called the **Inception module**

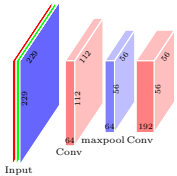


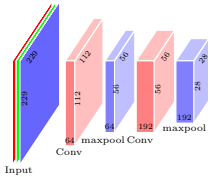
- But we might want to use different dimensionality reductions before the 3×3 and 5×5 filters
- So we can use D_1 and D_2 1×1 filters before the 3×3 and 5×5 filters respectively
- We can then add the maxpooling layer followed by dimensionality reduction
- And a new set of 1×1 convolutions
- And finally we concatenate all these layers
- This is called the **Inception module**
- We will now see **GoogLeNet** which contains many such inception modules

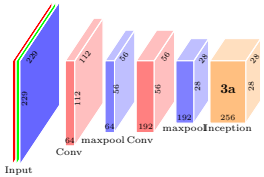

224
224
Input

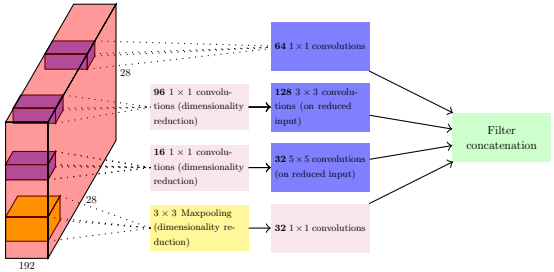

224
224
3
Input
112
112
64
Conv

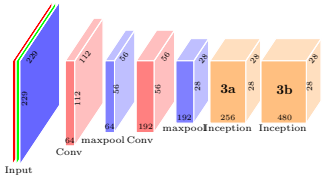

224
224
3
Input
112
112
64
Conv
56
56
64
maxpool

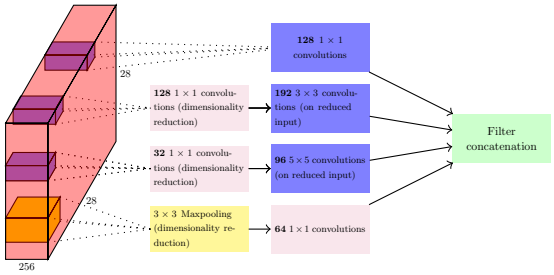

229
229
3
Input
112
112
64
Conv
56
56
64
maxpool
56
56
192
Conv

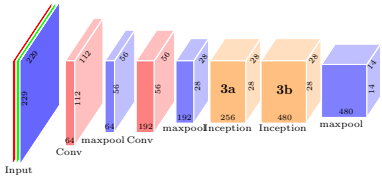

224
224
Input
112
112
64
Conv
56
56
64
maxpool
56
56
192
Conv
28
28
192
maxpool

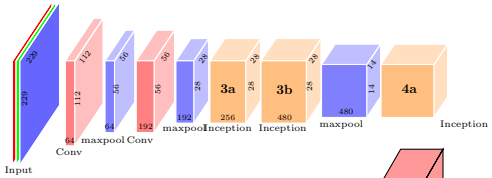

224
224
Input
112
112
64
Conv
56
56
64
maxpool
56
56
192
Conv
28
28
192
maxpool
3a
28
28
256
Inception

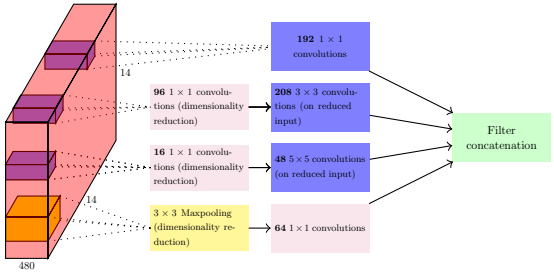

28
28
192
96 1 × 1 convolutions (dimensionality reduction)
16 1 × 1 convolutions (dimensionality reduction)
3 × 3 Maxpooling (dimensionality reduction)
64 1 × 1 convolutions
128 3 × 3 convolutions (on reduced input)
32 5 × 5 convolutions (on reduced input)
32 1 × 1 convolutions
Filter concatenation

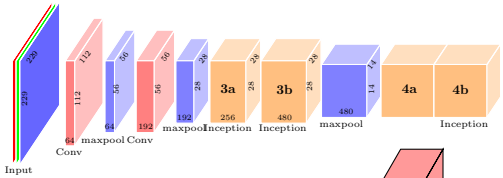

224
224
Input
112
112
64
Conv
56
56
64
maxpool
56
56
192
Conv
28
28
192
maxpool
28
28
3a
256
Inception
28
28
3b
480
Inception

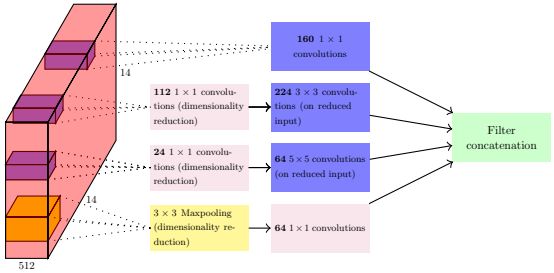

28
28
256
128 1 × 1 convolutions
128 1 × 1 convolutions (dimensionality reduction)
192 3 × 3 convolutions (on reduced input)
32 1 × 1 convolutions (dimensionality reduction)
96 5 × 5 convolutions (on reduced input)
3 × 3 Maxpooling (dimensionality reduction)
64 1 × 1 convolutions
Filter concatenation

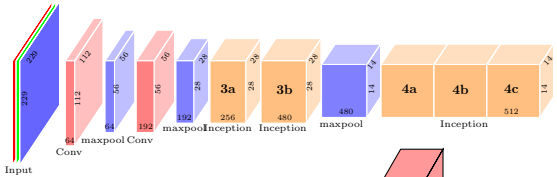

224
224
Input
112
112
64
Conv
56
56
64
maxpool
56
56
192
Conv
28
28
192
maxpool
28
28
3a
256
Inception
28
28
3b
480
Inception
14
14
480
maxpool

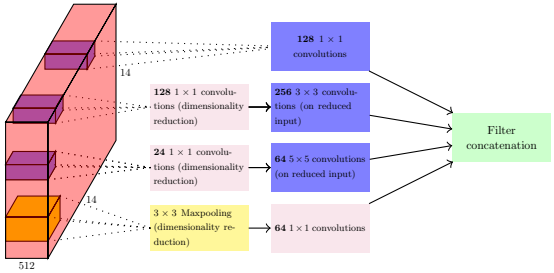

224
224
Input
112
112
64
Conv
56
56
64
maxpool
56
56
192
Conv
28
28
192
maxpool
28
28
3a
256
Inception
28
28
3b
480
Inception
28
28
480
maxpool
14
14
4a
Inception

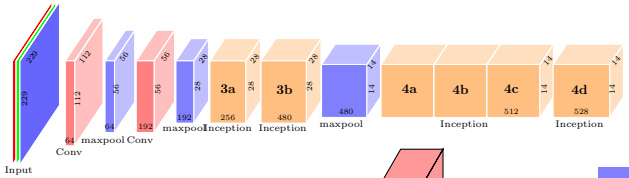

14
14
480
192 1 × 1 convolutions
96 1 × 1 convolutions (dimensionality reduction)
208 3 × 3 convolutions (on reduced input)
16 1 × 1 convolutions (dimensionality reduction)
48 5 × 5 convolutions (on reduced input)
3 × 3 Maxpooling (dimensionality reduction)
64 1 × 1 convolutions
Filter concatenation

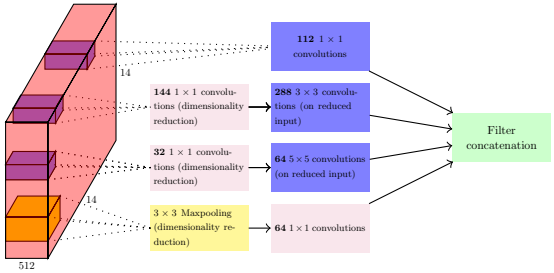

224
224
Input
112
112
64
Conv
56
56
64
maxpool
56
56
192
Conv
28
28
192
maxpool
28
28
3a
256
Inception
28
28
3b
480
Inception
28
28
480
maxpool
14
14
4a
Inception
4b

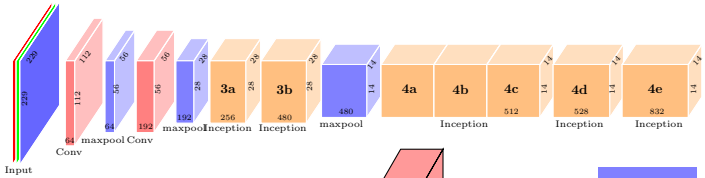

14
14
512
160 1 × 1 convolutions
112 1 × 1 convolutions (dimensionality reduction)
224 3 × 3 convolutions (on reduced input)
24 1 × 1 convolutions (dimensionality reduction)
64 5 × 5 convolutions (on reduced input)
3 × 3 Maxpooling (dimensionality reduction)
64 1 × 1 convolutions
Filter concatenation

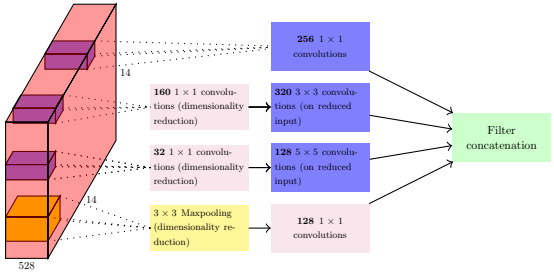

Input
224
224
Conv
112
112
64
maxpool
56
56
64
Conv
56
56
192
maxpool
28
28
192
Inception
3a
28
28
256
Inception
3b
28
28
480
maxpool
14
14
480
Inception
4a
4b
4c
14
14
512

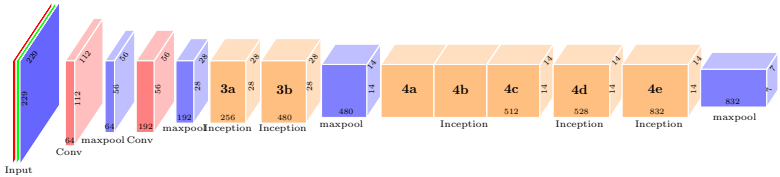

14
14
512
128 1 × 1 convolutions
128 1 × 1 convolutions (dimensionality reduction)
256 3 × 3 convolutions (on reduced input)
24 1 × 1 convolutions (dimensionality reduction)
64 5 × 5 convolutions (on reduced input)
3 × 3 Maxpooling (dimensionality reduction)
64 1 × 1 convolutions
Filter concatenation

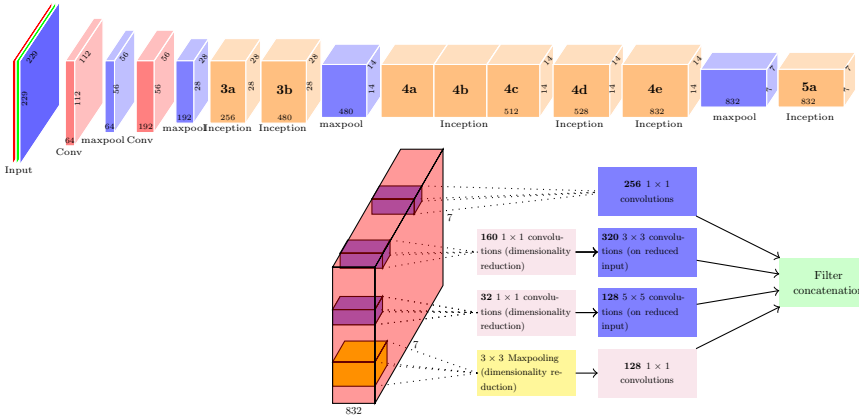

Input
Conv
maxpool
Conv
maxpool
Inception
Inception
maxpool
Inception
Inception

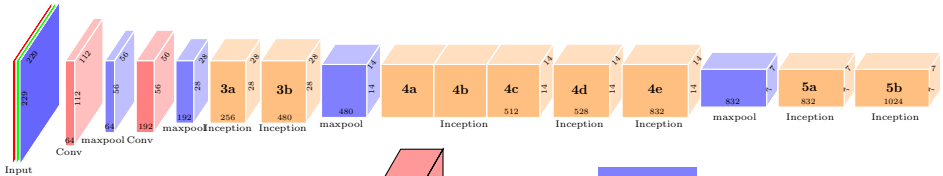

112 1 × 1 convolutions
144 1 × 1 convolutions (dimensionality reduction)
288 3 × 3 convolutions (on reduced input)
32 1 × 1 convolutions (dimensionality reduction)
64 5 × 5 convolutions (on reduced input)
3 × 3 Maxpooling (dimensionality reduction)
64 1 × 1 convolutions
Filter concatenation
14
14
512

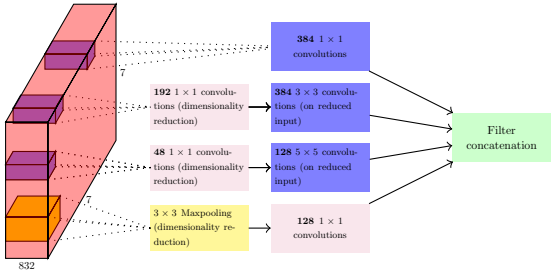

Input
224
224
Conv
112
112
64
maxpool
56
56
64
Conv
56
56
192
maxpool
28
28
192
Inception
3a
28
28
256
Inception
3b
28
28
480
maxpool
14
14
480
Inception
4a
4b
4c
14
14
512
Inception
4d
14
14
528
Inception
4e
14
14
832

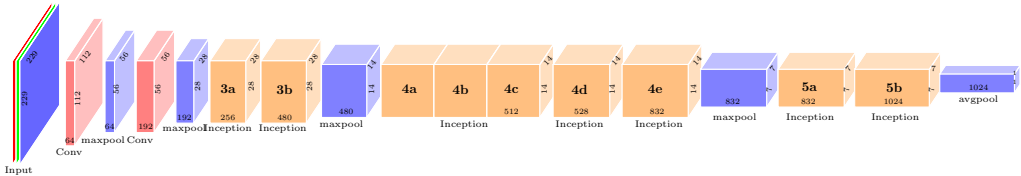

14
14
528
256 1 × 1 convolutions
160 1 × 1 convolutions (dimensionality reduction)
320 3 × 3 convolutions (on reduced input)
32 1 × 1 convolutions (dimensionality reduction)
128 5 × 5 convolutions (on reduced input)
3 × 3 Maxpooling (dimensionality reduction)
128 1 × 1 convolutions
Filter concatenation

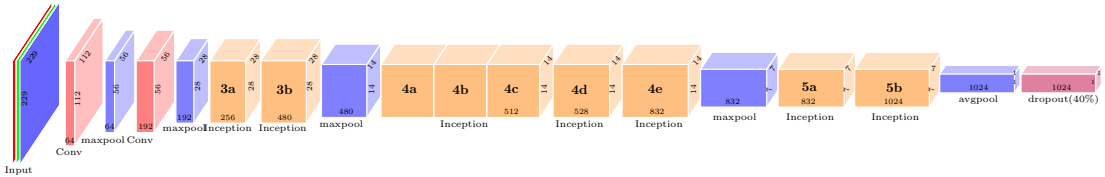

224
224
Input
112
112
64
Conv
56
56
64
maxpool
56
56
192
Conv
28
28
192
maxpool
3a
28
28
256
Inception
3b
28
28
480
Inception
28
14
14
480
maxpool
4a
4b
4c
14
14
512
Inception
4d
14
14
528
Inception
4e
14
14
832
Inception
7
7
832
maxpool

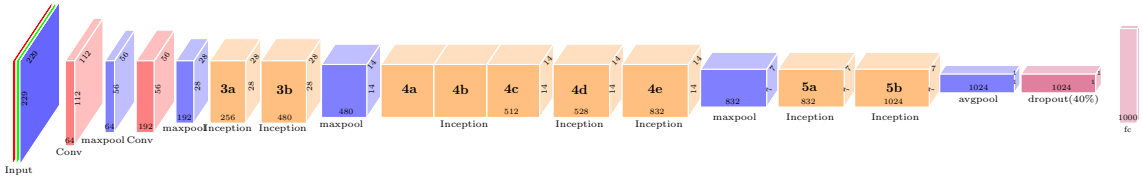

Input
Conv
maxpool
Conv
maxpool
Inception
3a
Inception
3b
maxpool
4a
Inception
4b
4c
Inception
4d
Inception
4e
maxpool
Inception
5a
256 1 × 1 convolutions
160 1 × 1 convolutions (dimensionality reduction)
320 3 × 3 convolutions (on reduced input)
32 1 × 1 convolutions (dimensionality reduction)
128 5 × 5 convolutions (on reduced input)
3 × 3 Maxpooling (dimensionality reduction)
128 1 × 1 convolutions
Filter concatenation
832

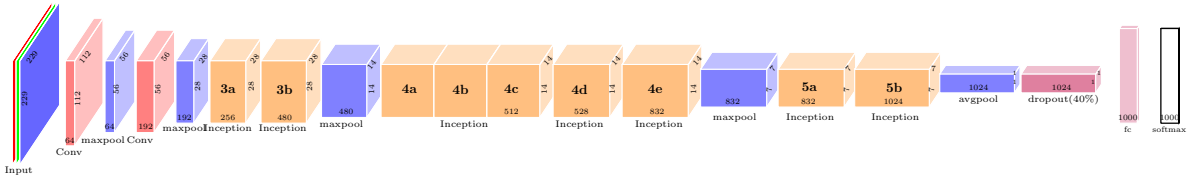

Input
224
224
Conv
112
112
64
maxpool
56
56
64
Conv
56
56
192
maxpool
28
28
192
Inception
3a
28
28
256
Inception
3b
28
28
480
maxpool
14
14
480
4a
4b
4c
14
14
512
Inception
4d
14
14
528
Inception
4e
14
14
832
maxpool
7
7
832
5a
7
7
832
Inception
5b
7
7
1024
Inception


7
832
384 1 × 1 convolutions
192 1 × 1 convolutions (dimensionality reduction)
384 3 × 3 convolutions (on reduced input)
48 1 × 1 convolutions (dimensionality reduction)
128 5 × 5 convolutions (on reduced input)
3 × 3 Maxpooling (dimensionality reduction)
128 1 × 1 convolutions
Filter concatenation

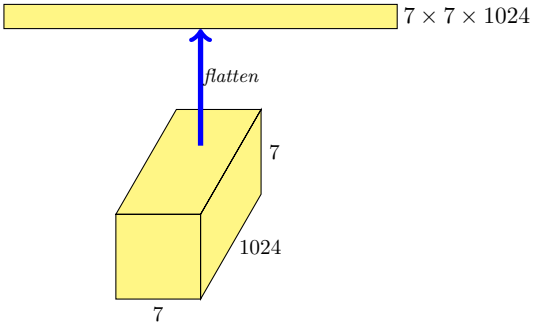

Input
224
224
Conv
112
112
64
maxpool
56
56
64
Conv
56
56
192
maxpool
28
28
192
Inception
3a
28
28
256
Inception
3b
28
28
480
maxpool
14
14
480
Inception
4a
4b
4c
14
14
512
Inception
4d
14
14
528
Inception
4e
14
14
832
maxpool
7
7
832
Inception
5a
7
7
832
Inception
5b
7
7
1024
avgpool
1
1
1024


224
224
Input
112
112
64
Conv
56
56
64
maxpool
56
56
192
Conv
28
28
192
maxpool
3a
28
28
256
Inception
3b
28
28
480
Inception
28
14
480
maxpool
4a
4b
4c
14
14
512
Inception
4d
14
14
528
Inception
4e
14
14
832
Inception
7
7
832
maxpool
5a
7
7
832
Inception
5b
7
7
1024
Inception
1
1
1024
avgpool
1
1
1024
dropout(40%)

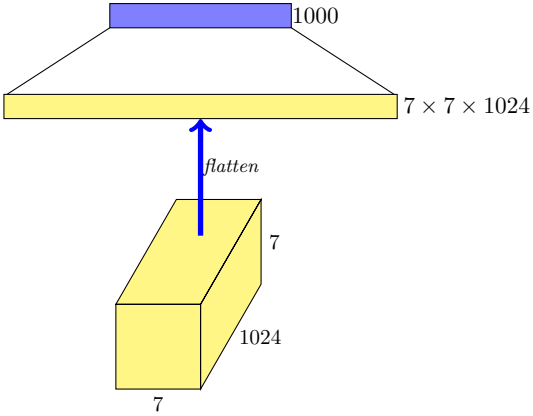

Input
Conv
maxpool
Conv
maxpool
Inception
Inception
maxpool
Inception
Inception
Inception
Inception
maxpool
Inception
Inception
avgpool
dropout(40%)
fc


Input
Conv
maxpool
Conv
maxpool
Inception
Inception
maxpool
Inception
Inception
Inception
maxpool
Inception
Inception
avgpool
dropout(40%)
fc
softmax

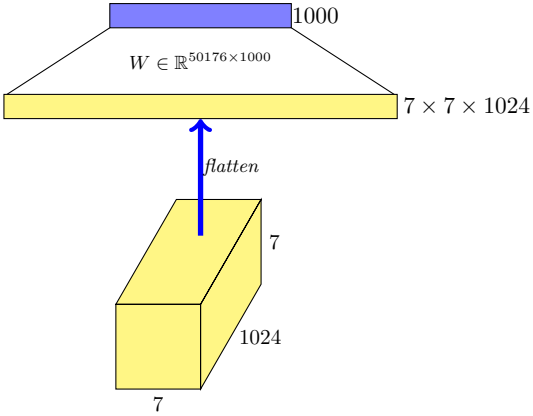
- **Important Trick:** Got rid of the fully connected layer



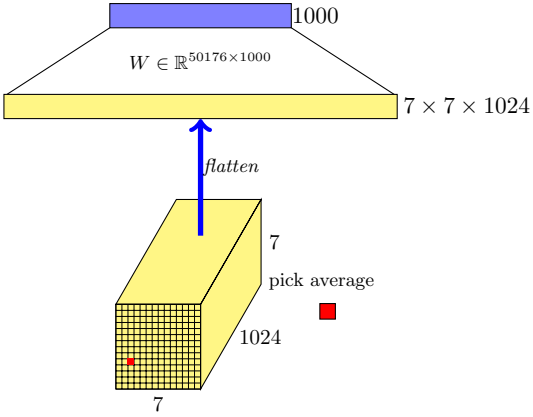
- **Important Trick:** Got rid of the fully connected layer
- Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional



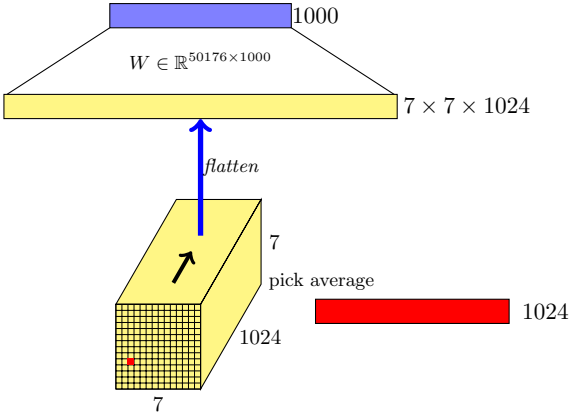
- **Important Trick:** Got rid of the fully connected layer
- Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional
- What if we were to add a fully connected layer with 1000 nodes (for 1000 classes) on top of this



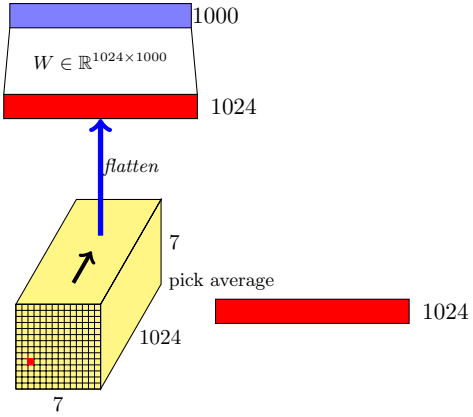
- **Important Trick:** Got rid of the fully connected layer
- Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional
- What if we were to add a fully connected layer with 1000 nodes (for 1000 classes) on top of this
- We would have $7 \times 7 \times 1024 \times 1000 = 49M$ *parameters*



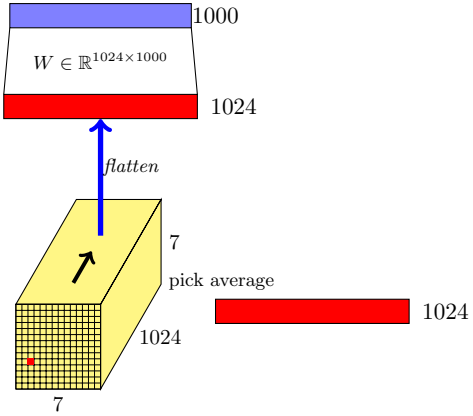
- **Important Trick:** Got rid of the fully connected layer
- Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional
- What if we were to add a fully connected layer with 1000 nodes (for 1000 classes) on top of this
- We would have $7 \times 7 \times 1024 \times 1000 = 49M$ *parameters*
- Instead they use an average pooling of size 7×7 on each of the 1024 feature maps



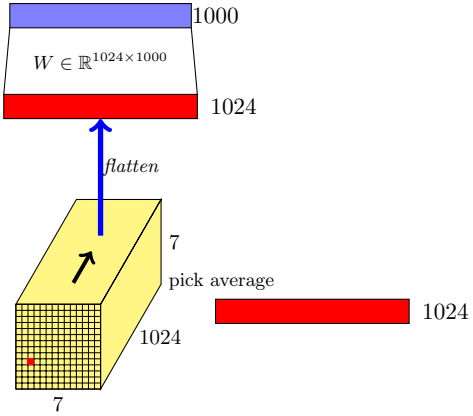
- **Important Trick:** Got rid of the fully connected layer
- Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional
- What if we were to add a fully connected layer with 1000 nodes (for 1000 classes) on top of this
- We would have $7 \times 7 \times 1024 \times 1000 = 49M$ *parameters*
- Instead they use an average pooling of size 7×7 on each of the 1024 feature maps
- This results in a 1024 dimensional output



- **Important Trick:** Got rid of the fully connected layer
- Notice that output of the last layer is $7 \times 7 \times 1024$ dimensional
- What if we were to add a fully connected layer with 1000 nodes (for 1000 classes) on top of this
- We would have $7 \times 7 \times 1024 \times 1000 = 49M$ *parameters*
- Instead they use an average pooling of size 7×7 on each of the 1024 feature maps
- This results in a 1024 dimensional output
- Significantly reduces the number of parameters

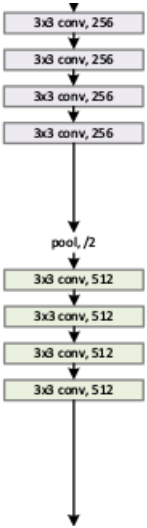


- 12× less parameters than AlexNet

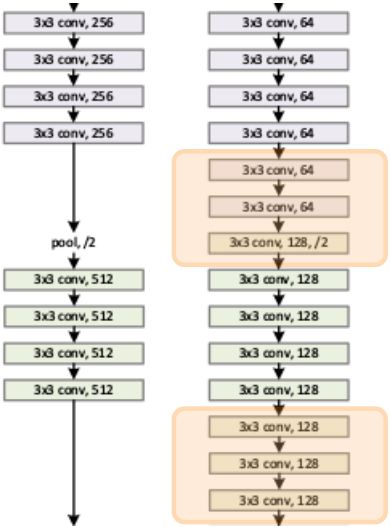


- 12× less parameters than AlexNet
- 2× more computations

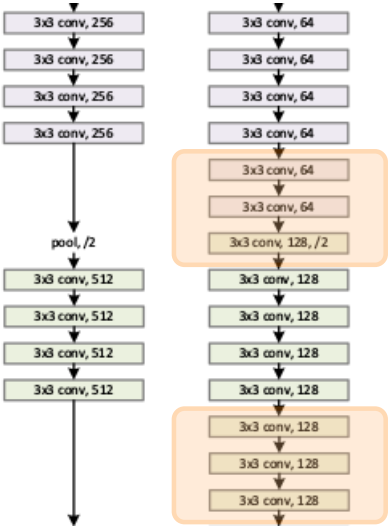
- GoogLeNet
- ResNet



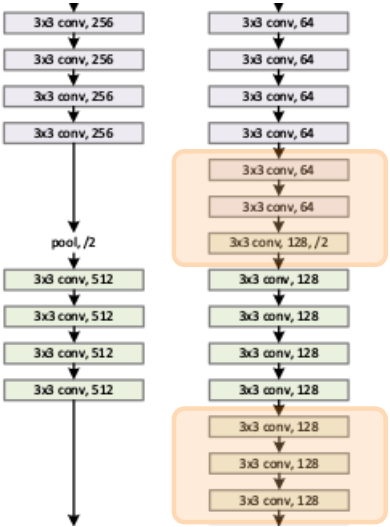
- Suppose we have been able to train a shallow neural network well



- Suppose we have been able to train a shallow neural network well
- Now suppose we construct a deeper network which has few more layers (in orange)

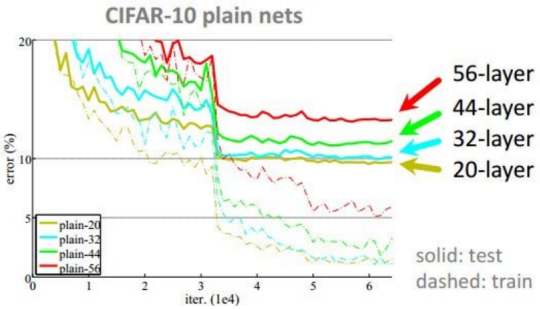


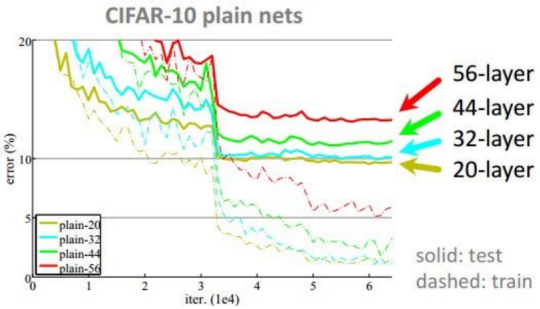
- Suppose we have been able to train a shallow neural network well
- Now suppose we construct a deeper network which has few more layers (in orange)
- Intuitively, if the shallow network works well then the deep network should also work well by simply learning to compute identity functions in the new layers



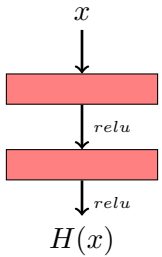
- Suppose we have been able to train a shallow neural network well
- Now suppose we construct a deeper network which has few more layers (in orange)
- Intuitively, if the shallow network works well then the deep network should also work well by simply learning to compute identity functions in the new layers
- Essentially, the solution space of a shallow neural network is a subset of the solution space of a deep neural network

- But in practice it is observed that this doesn't happen

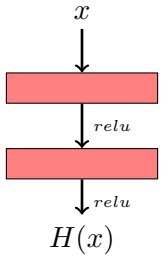




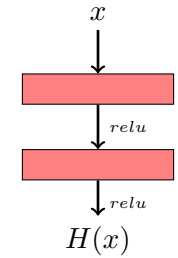
- But in practice it is observed that this doesn't happen
- Notice that the deep layers have a higher error rate on the test set

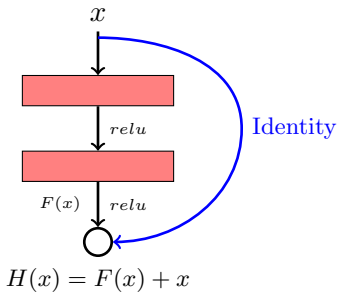


- Consider any two stacked layers in a CNN

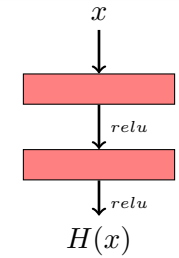


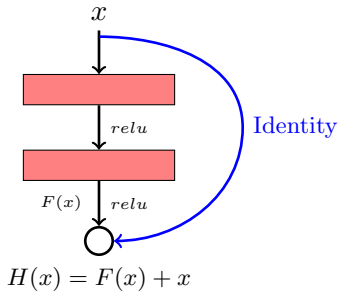
- Consider any two stacked layers in a CNN
- The two layers are essentially learning some function of the input



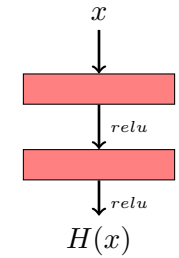


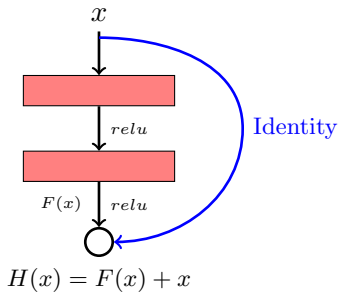
- Consider any two stacked layers in a CNN
- The two layers are essentially learning some function of the input
- What if we enable it to learn only a residual function of the input?



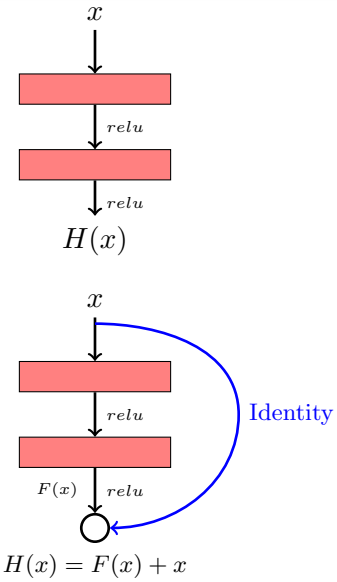


- Why would this help?

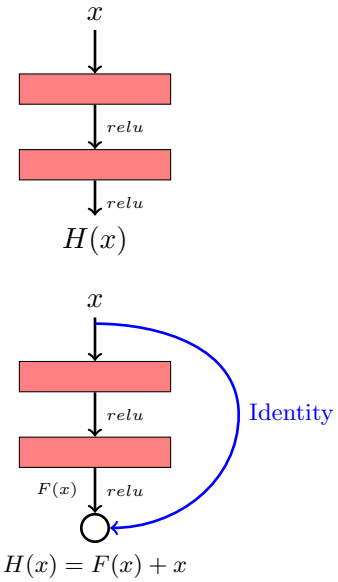




- Why would this help?
- Remember our argument that a deeper version of a shallow network would do just fine by learning identity transformations in the new layers



- Why would this help?
- Remember our argument that a deeper version of a shallow network would do just fine by learning identity transformations in the new layers
- This identity connection from the input allows a ResNet to retain a copy of the input



- Why would this help?
- Remember our argument that a deeper version of a shallow network would do just fine by learning identity transformations in the new layers
- This identity connection from the input allows a ResNet to retain a copy of the input
- Using this idea they were able to train really deep networks

ResNet, 152 layers

1^{st} place in all five main tracks

- **ImageNet Classification:** "Ultra-deep" 152-layer nets

ResNet, 152 layers

1^{st} place in all five main tracks

- **ImageNet Classification:** "Ultra-deep" 152-layer nets
- **ImageNet Detection:** 16% better than the 2nd best system

ResNet, 152 layers

1^{st} place in all five main tracks

- **ImageNet Classification:** "Ultra-deep" 152-layer nets
- **ImageNet Detection:** 16% better than the 2nd best system
- **ImageNet Localization:** 27% better than the 2nd best system

ResNet, 152 layers

1^{st} place in all five main tracks

- **ImageNet Classification:** "Ultra-deep" 152-layer nets
- **ImageNet Detection:** 16% better than the 2nd best system
- **ImageNet Localization:** 27% better than the 2nd best system
- **COCO Detection:** 11% better than the 2nd best system

ResNet, 152 layers

1^{st} place in all five main tracks

- **ImageNet Classification:** "Ultra-deep" 152-layer nets
- **ImageNet Detection:** 16% better than the 2nd best system
- **ImageNet Localization:** 27% better than the 2nd best system
- **COCO Detection:** 11% better than the 2nd best system
- **COCO Segmentation:** 12% better than the 2nd best system

ResNet, 152 layers

Bag of tricks

- Batch Normalizaton after every CONV layer

ResNet, 152 layers

Bag of tricks

- Batch Normalizaton after every CONV layer
- Xavier/2 initialization from [He et al]

ResNet, 152 layers

Bag of tricks

- Batch Normalizaton after every CONV layer
- Xavier/2 initialization from [He et al]
- SGD + Momentum(0.9)

ResNet, 152 layers

Bag of tricks

- Batch Normalizaton after every CONV layer
- Xavier/2 initialization from [He et al]
- SGD + Momentum(0.9)
- Learning rate:0.1, divided by 10 when validation error plateaus

ResNet, 152 layers

Bag of tricks

- Batch Normalizaton after every CONV layer
- Xavier/2 initialization from [He et al]
- SGD + Momentum(0.9)
- Learning rate:0.1, divided by 10 when validation error plateaus
- Mini-batch size 256

ResNet, 152 layers

Bag of tricks

- Batch Normalizaton after every CONV layer
- Xavier/2 initialization from [He et al]
- SGD + Momentum(0.9)
- Learning rate:0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5

ResNet, 152 layers

Bag of tricks

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