Week 6: Deep Learning

From CS231, 2017, Stanford

Sciences U Lyon

Computer Vision Challenges

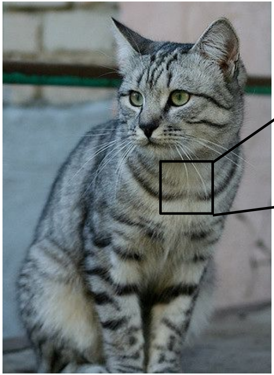
Image Classification: A core task in Computer Vision

This image by Nikita is licensed under CC-BY 2.0

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

The Problem: Semantic Gap



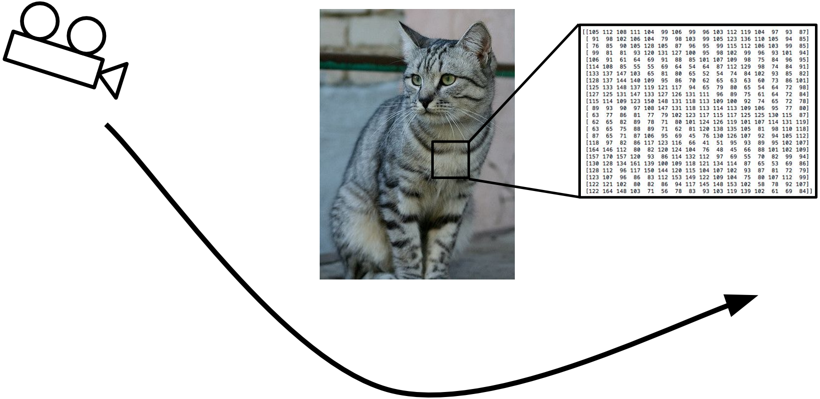

```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Challenges: Viewpoint variation



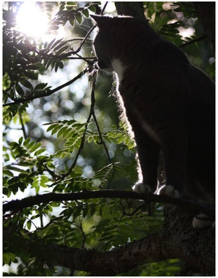
All pixels change when the camera moves!

Challenges: Illumination

This image is CC0 1.0 public domain

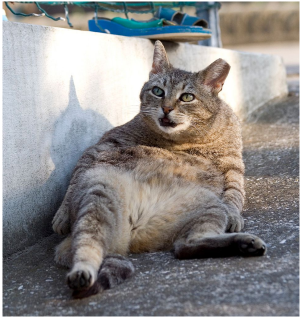
This image is CC0 1.0 public domain

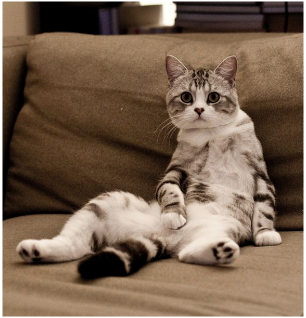
This image is CC0 1.0 public domain



This image is CC0 1.0 public domain

Challenges: Deformation





Challenges: Occlusion

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image by jonsson is licensed under CC-BY 2.0

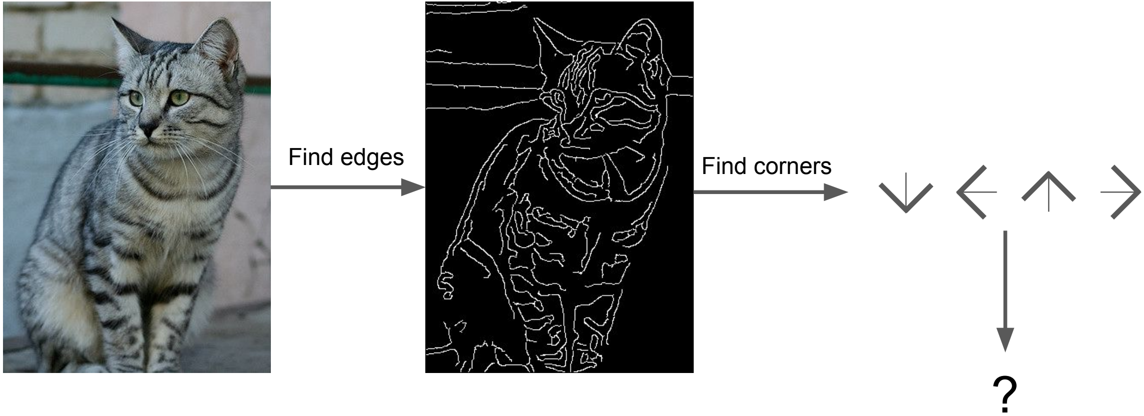
Challenges: Background Clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Challenges: Intraclass variation

Attempts have been made



John Canny, "A Computational Approach to Edge Detection", IEEE TPAMI 1986

Data-Driven Approach

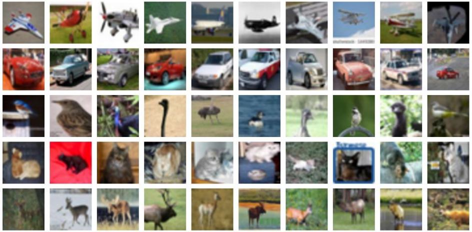
1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

```
def train(images, labels):
  # Machine learning!
  return model
```

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Example training set

airplane
automobile
bird
cat
deer



First classifier: **Nearest Neighbor**

```
def train(images, labels):
  # Machine learning!
  return model
```

→ Memorize all data and labels

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

→ Predict the label of the most similar training image

Example Dataset: **CIFAR10**

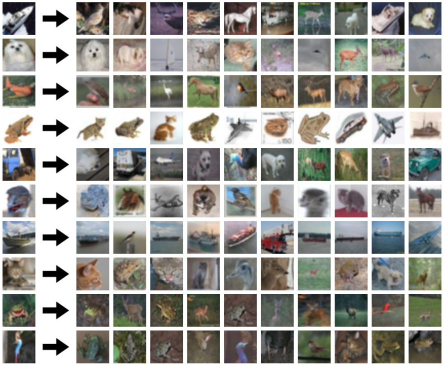
10 classes
50,000 training images
10,000 testing images

Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

Example Dataset: **CIFAR10**

10 classes
50,000 training images
10,000 testing images

Test images and nearest neighbors



Alex Krizhevsky, “Learning Multiple Layers of Features from Tiny Images”, Technical Report, 2009.

k-Nearest Neighbor on images **never used.**

- Very slow at test time
- Distance metrics on pixels are not informative

Original

Boxed

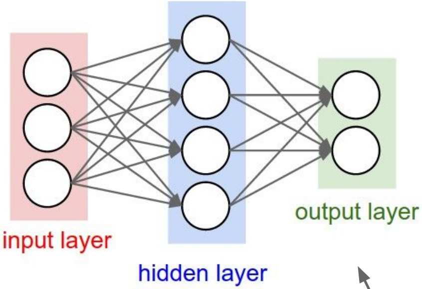
Shifted

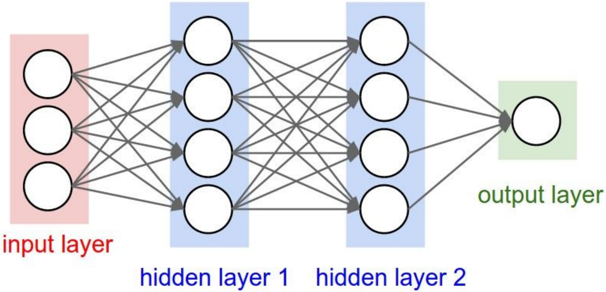
Tinted

(all 3 images have same L2 distance to the one on the left)

Neural Networks (NN)

Neural networks: Architectures





“2-layer Neural Net”, or
“1-hidden-layer Neural Net”

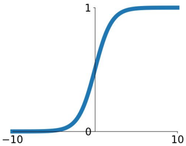
“3-layer Neural Net”, or
“2-hidden-layer Neural Net”

“Fully-connected” layers

Activation functions

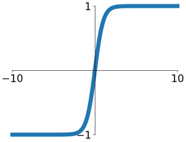
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



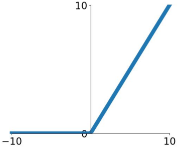
tanh

$\tanh(x)$



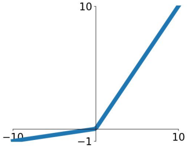
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

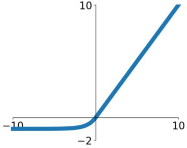


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

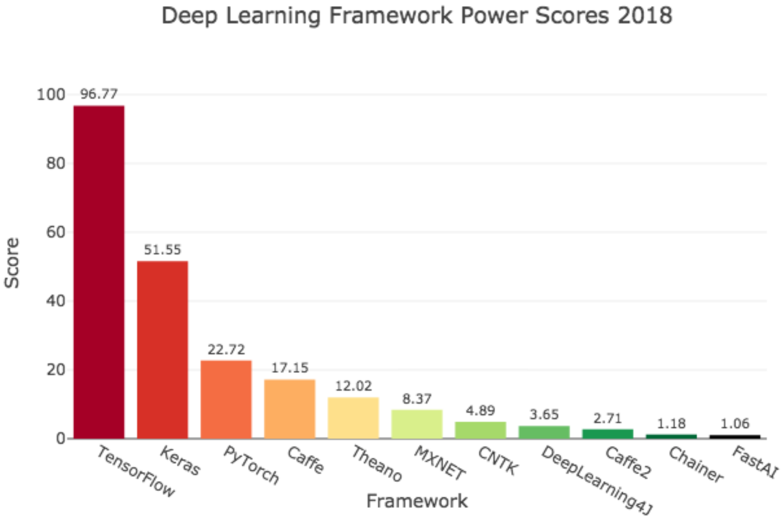


Keras Feed-forward Neural Network

```python
model = Sequential()

#layer 1:
model.add(Dense(100, input_dim=200, activation='relu'))

#layer 2:
model.add(Dense(50, activation='relu'))

#output layer:
model.add(Dense(5, activation='softmax'))
```

Deep Learning libraries

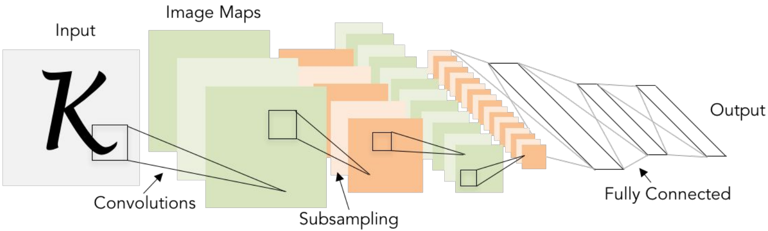
Deep Learning libraries



Convolutional Neural Networks (CNN)

A bit of history:

Gradient-based learning applied to document recognition
[LeCun, Bottou, Bengio, Haffner 1998]

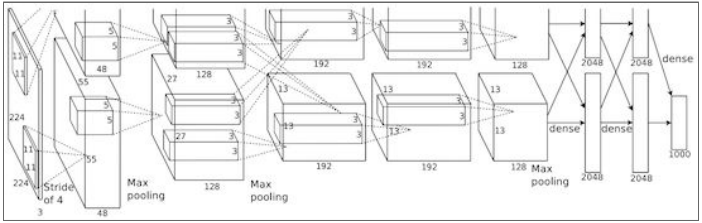


LeNet-5

A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

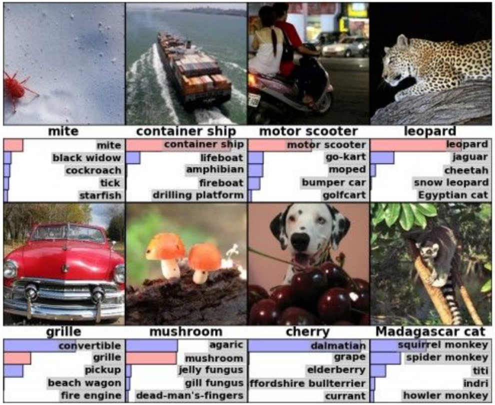
[Krizhevsky, Sutskever, Hinton, 2012]



"AlexNet"

Fast-forward to today: ConvNets are everywhere

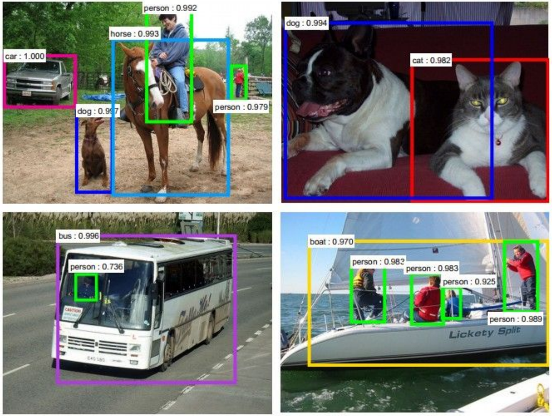
Classification



Retrieval

Fast-forward to today: ConvNets are everywhere

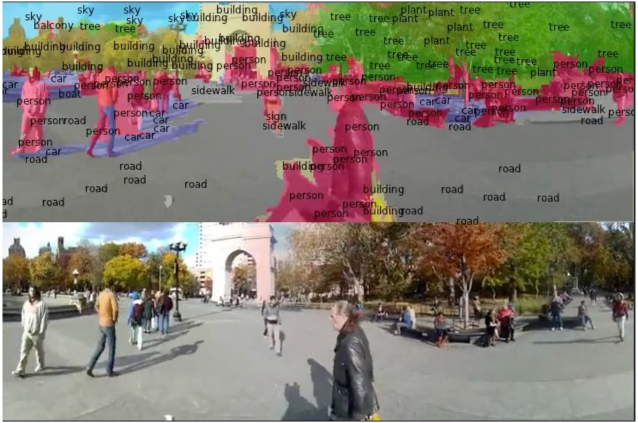
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

Image Captioning

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

No errors

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

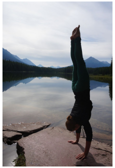
Minor errors

A man in a baseball uniform throwing a ball

A cat sitting on a suitcase on the floor

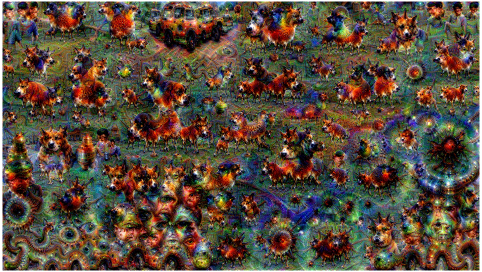
Somewhat related

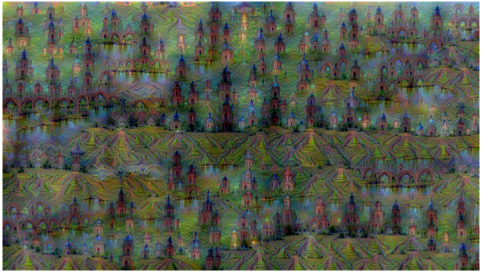
A woman is holding a cat in her hand



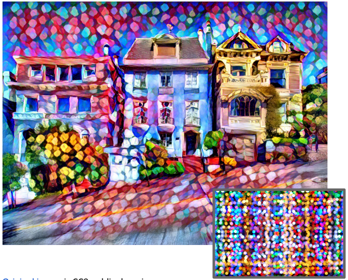
A woman standing on a beach holding a surfboard

Captions generated by Justin Johnson using Neuraltalk2





Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.



Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;
reproduced with permission

Gatys et al, “Image Style Transfer using Convolutional Neural Networks”, CVPR 2016
Gatys et al, “Controlling Perceptual Factors in Neural Style Transfer”, CVPR 2017

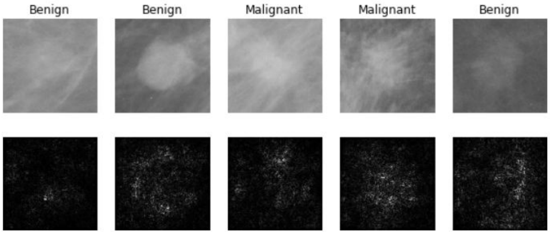
Fast-forward to today: ConvNets are everywhere

self-driving cars

NVIDIA Tesla line
(these are the GPUs on rye01.stanford.edu)

Note that for embedded systems a typical setup would involve NVIDIA Tegras, with integrated GPU and ARM-based CPU cores.

Fast-forward to today: ConvNets are everywhere



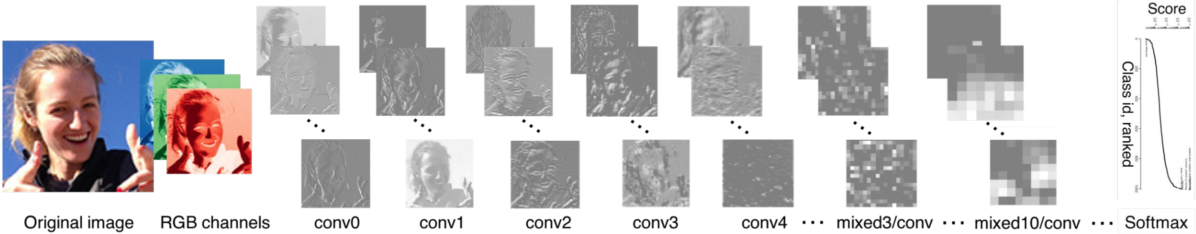
[Levy et al. 2016]

[Dieleman et al. 2014]

[Sermanet et al. 2011]
[Ciresan et al.]

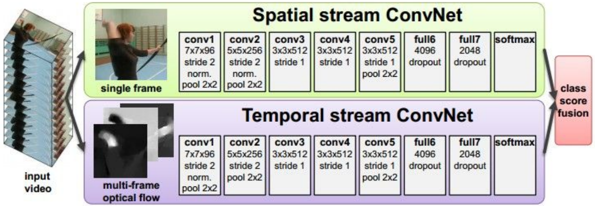
SO MUCH OF "AI" IS JUST FIGURING OUT WAYS TO OFFLOAD WORK ONTO RANDOM STRANGERS.

Fast-forward to today: ConvNets are everywhere



[Taigman et al. 2014]

Activations of inception-v3 architecture [Szegedy et al. 2015] to image of Emma McIntosh, used with permission. Figure and architecture not from Taigman et al. 2014.



[Simonyan et al. 2014]

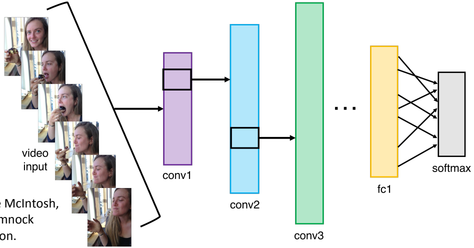
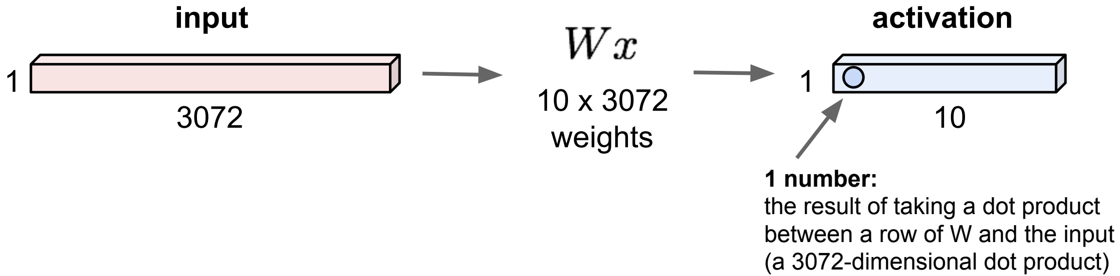


Illustration by Lane McIntosh, photos of Katie Cumnock used with permission.

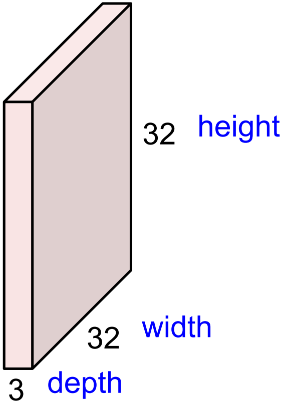
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



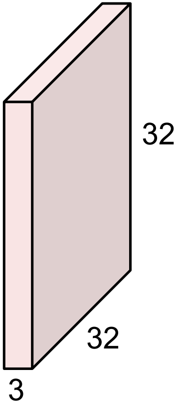
Convolution Layer

32x32x3 image -> preserve spatial structure

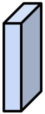


Convolution Layer

32x32x3 image



5x5x3 filter

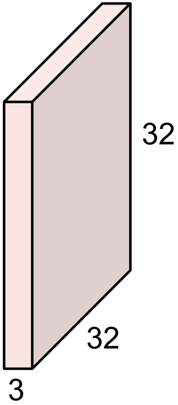


Convolve the filter with the image
i.e. "slide over the image spatially, computing dot products"

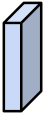
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

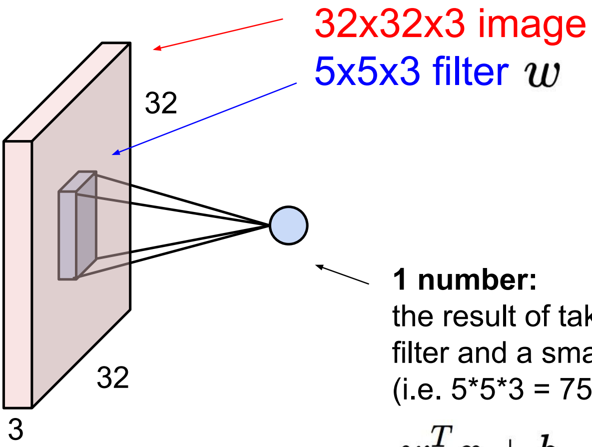


5x5x3 filter



Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

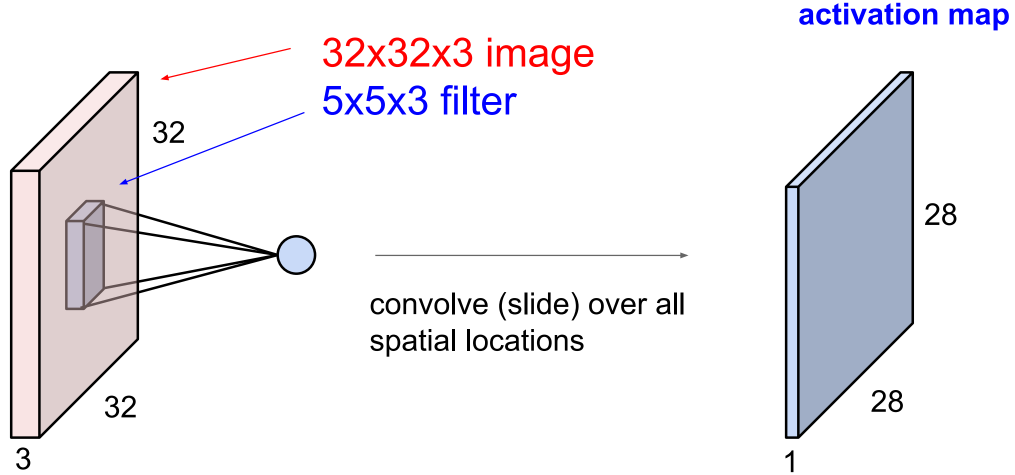
Convolution Layer



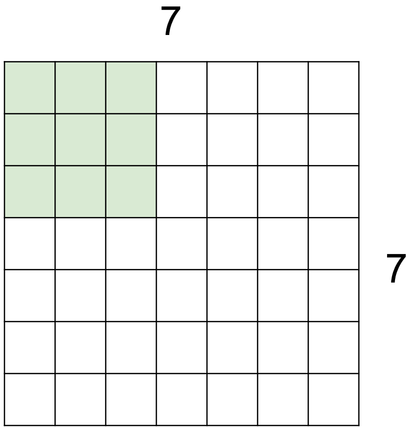
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolution Layer

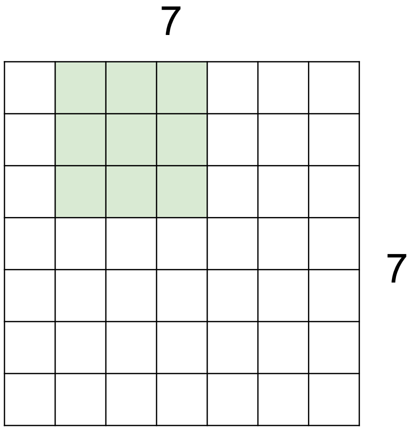


A closer look at spatial dimensions:



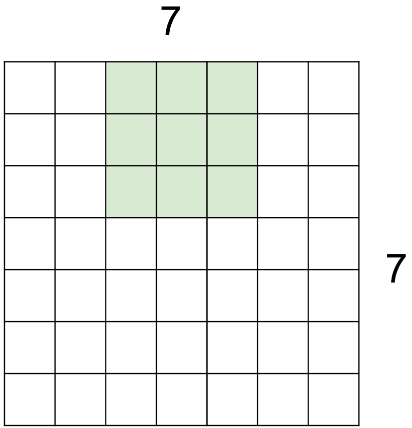
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



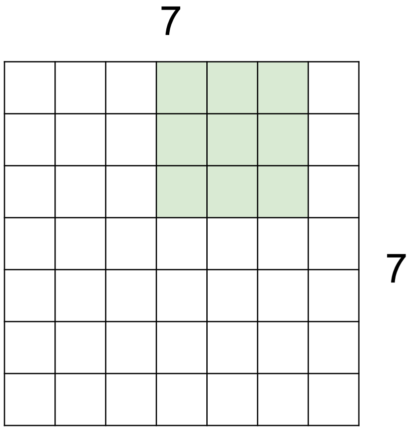
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



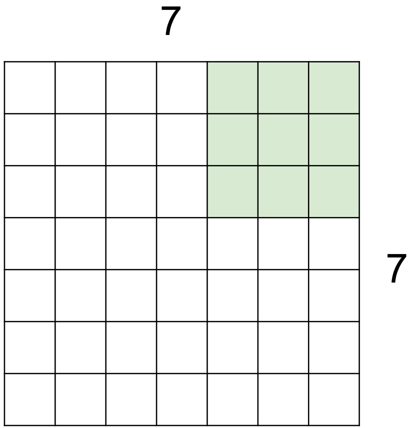
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:

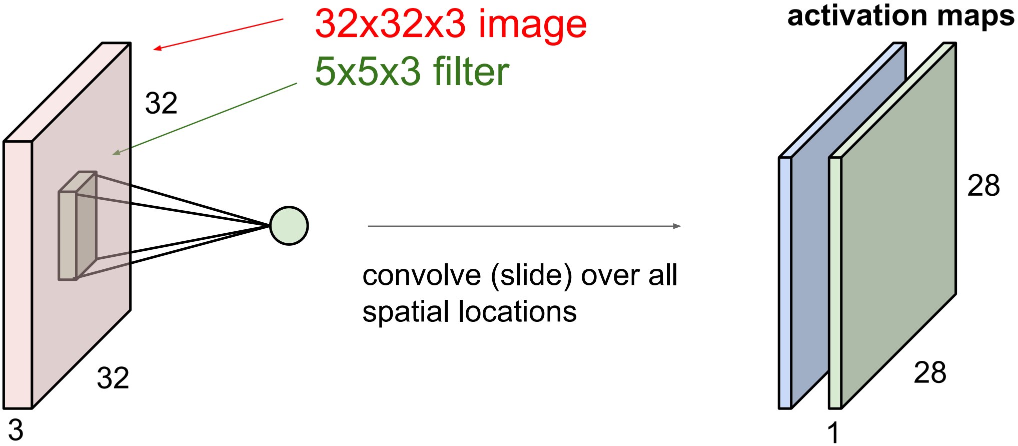


7x7 input (spatially)
assume 3x3 filter

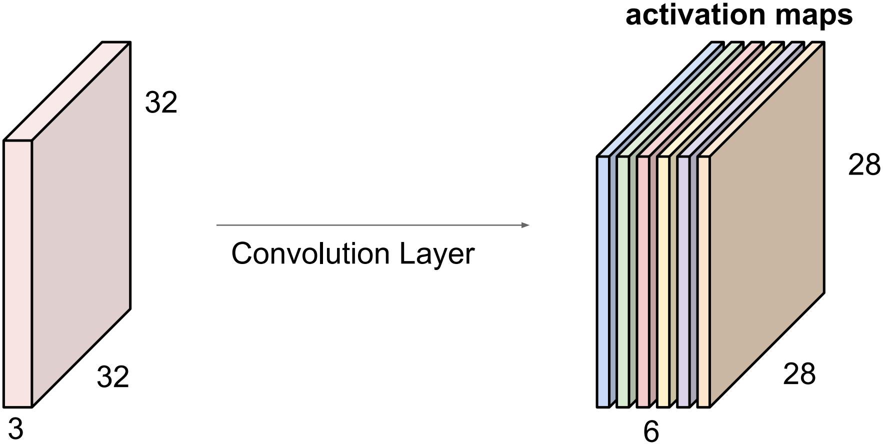
=> 5x5 output

Convolution Layer

consider a second, green filter

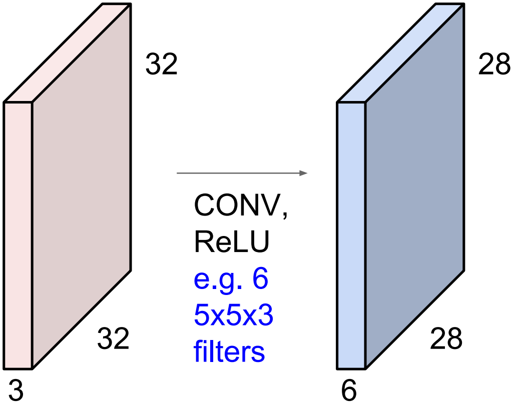


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

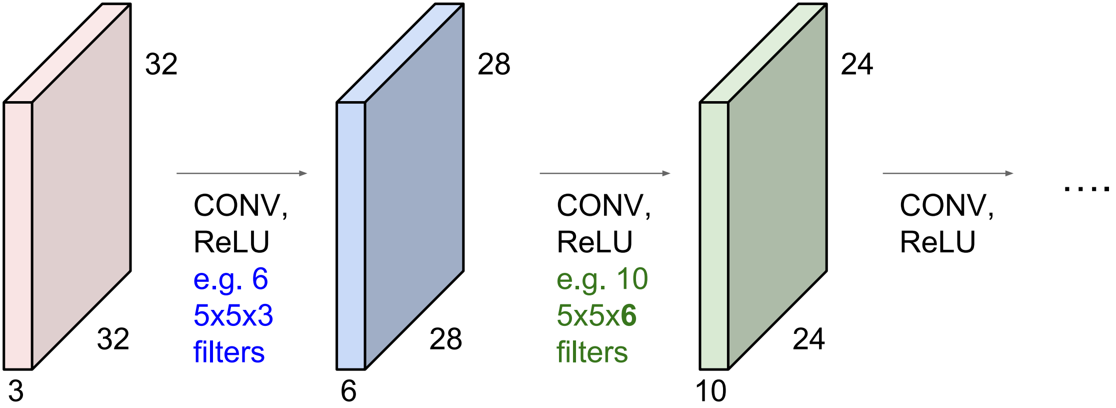


We stack these up to get a "new image" of size 28x28x6!

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



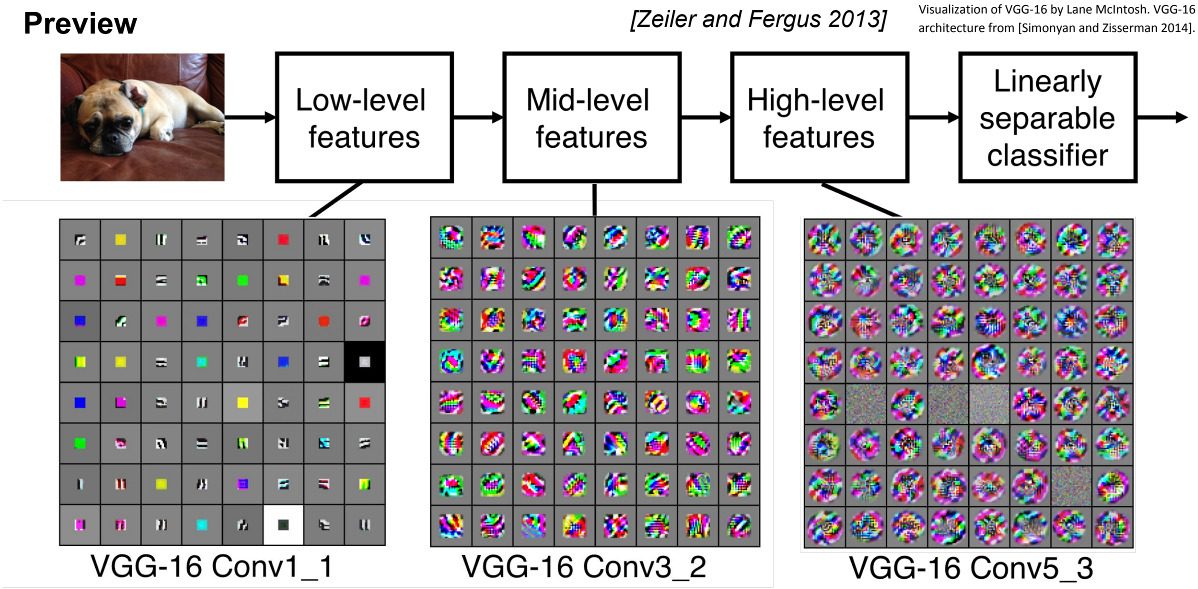
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



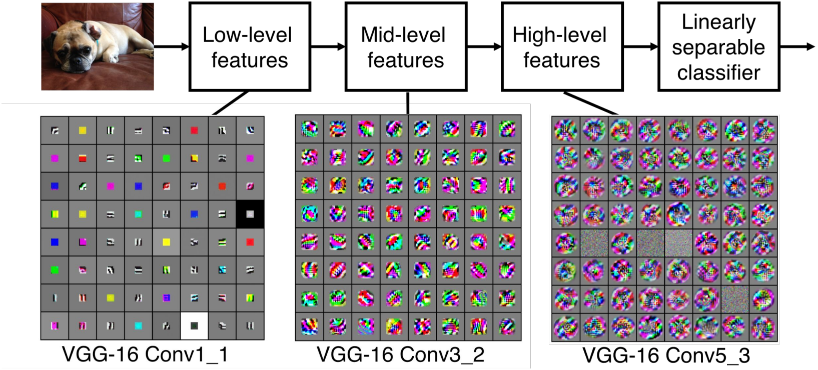
Keras Convolutional Neural Network

```python
model = Sequential()

model.add(Conv2D(6, (5, 5), activation='relu', input_shape=(32, 32, 3)))

model.add(Conv2D(10, (5, 5), activation='relu'))
```

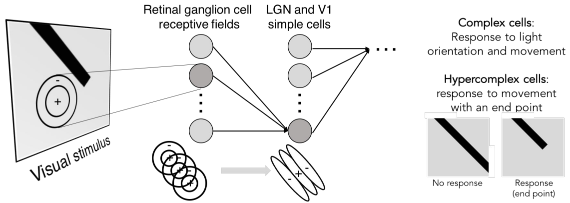
Preview


[Zeiler and Fergus 2013]
Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].
Low-level features
Mid-level features
High-level features
Linearly separable classifier
VGG-16 Conv1_1
VGG-16 Conv3_2
VGG-16 Conv5_3

Preview

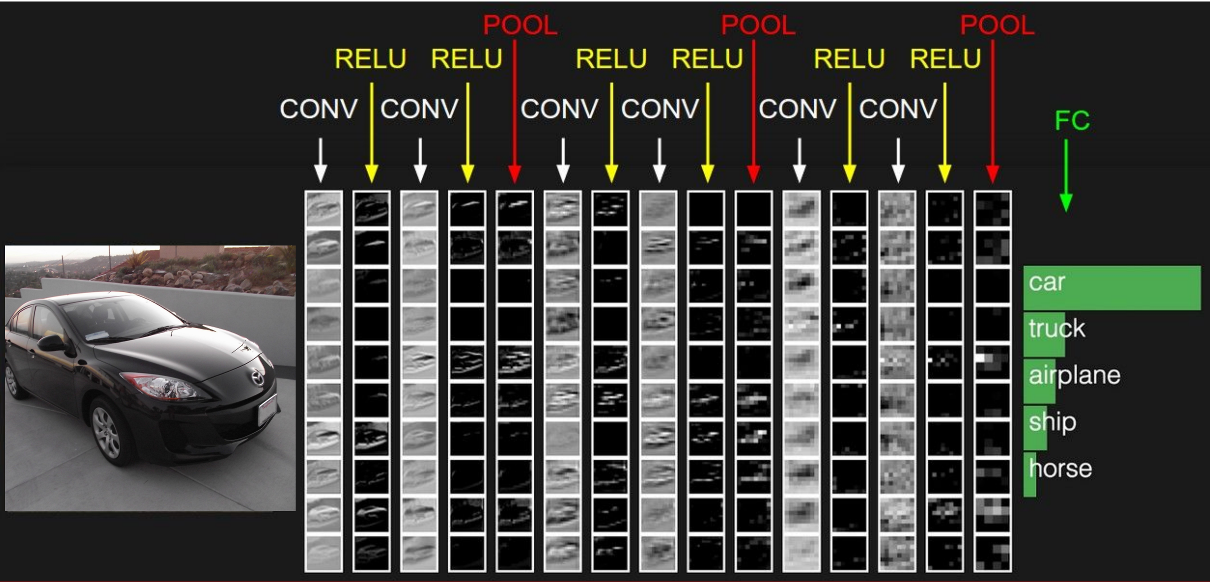


Low-level features
Mid-level features
High-level features
Linearly separable classifier
VGG-16 Conv1_1
VGG-16 Conv3_2
VGG-16 Conv5_3



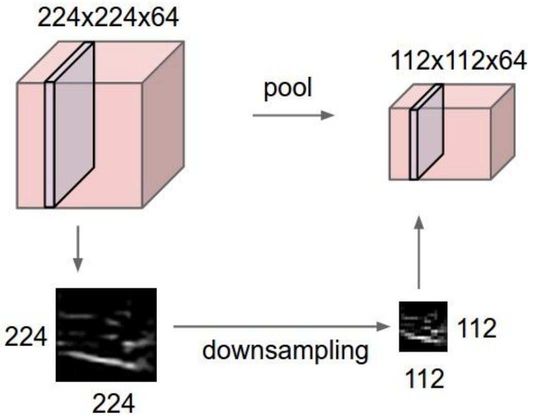
Retinal ganglion cell receptive fields
LGN and V1 simple cells
Visual stimulus
Complex cells:
Response to light orientation and movement
Hypercomplex cells:
response to movement with an end point
No response
Response (end point)

preview:

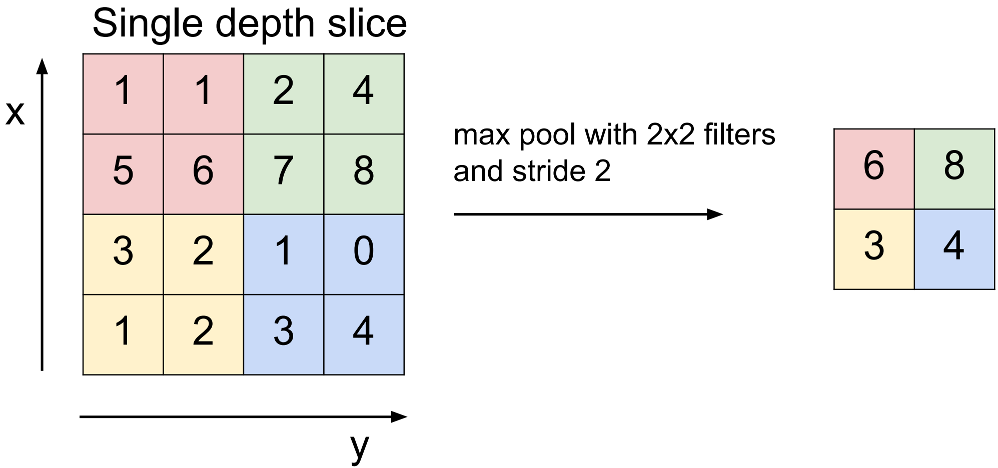

POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



MAX POOLING


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Keras Convolutional Neural Network

```python
model = Sequential()

model.add(Conv2D(6, (5, 5), activation='relu', input_shape=(32, 32, 3)))

model.add(Conv2D(10, (5, 5), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
```

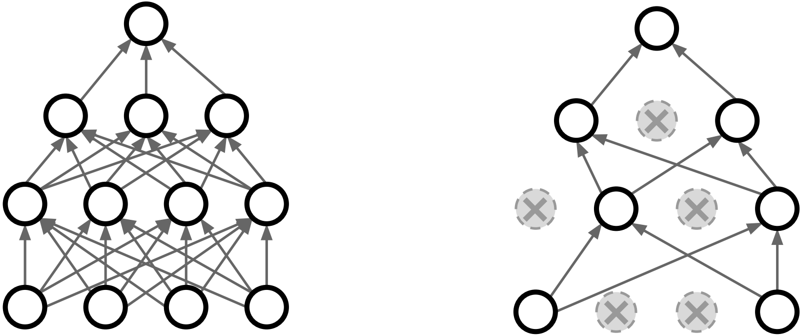
Hyperparameters to play with:

- network architecture
- learning rate, its decay schedule, update type
- regularization (L2/Dropout strength)

neural networks practitioner
music = loss function

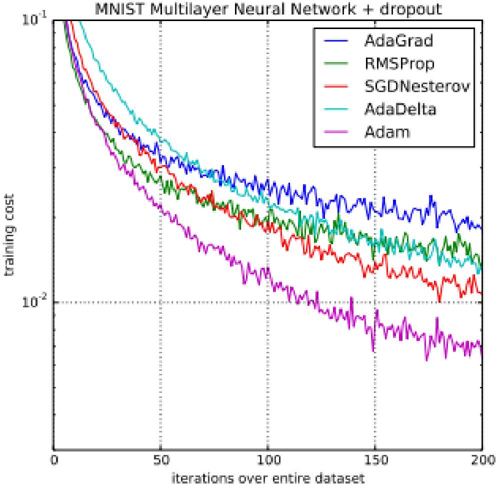
Regularization: Dropout

In each forward pass, randomly set some neurons to zero
Probability of dropping is a hyperparameter; 0.5 is common



Srivastava et al, "Dropout: A simple way to prevent neural networks from overfitting", JMLR 2014

Optimizers



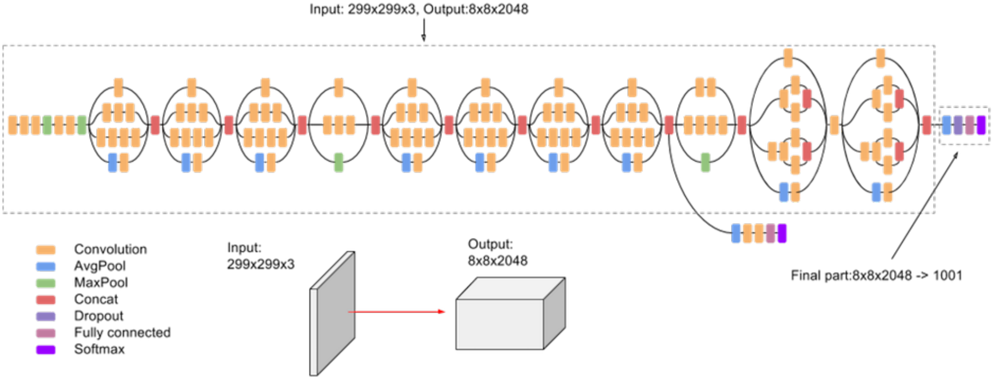
MNIST Multilayer Neural Network + dropout
10^{-1}
10^{-2}
training cost
0
50
100
150
200
iterations over entire dataset
AdaGrad
RMSProp
SGDNesterov
AdaDelta
Adam

Keras Full Convolutional Neural Network

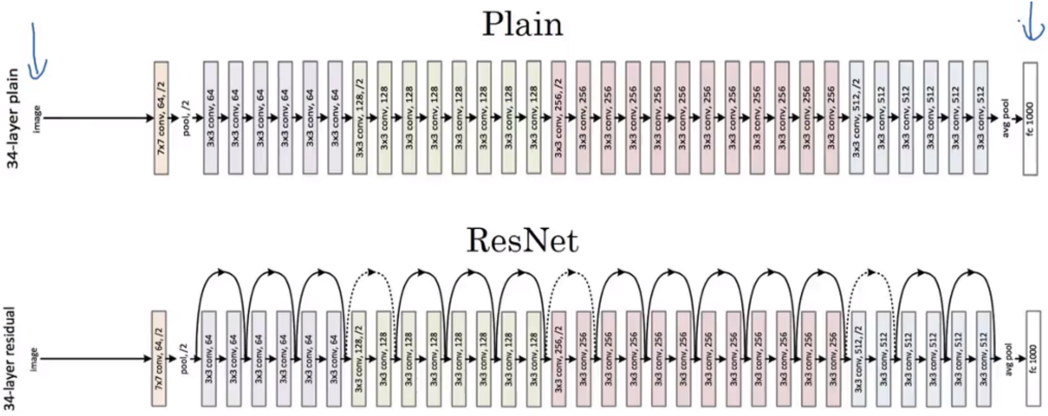
```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 3)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))


model.compile(loss='categorical_crossentropy', optimizer=Adam(lr = 0.001))

model.fit(x_train, y_train, batch_size=32, epochs=10)
score = model.evaluate(x_test, y_test, batch_size=32)
```

State-of-the-art
Neural Networks Architectures

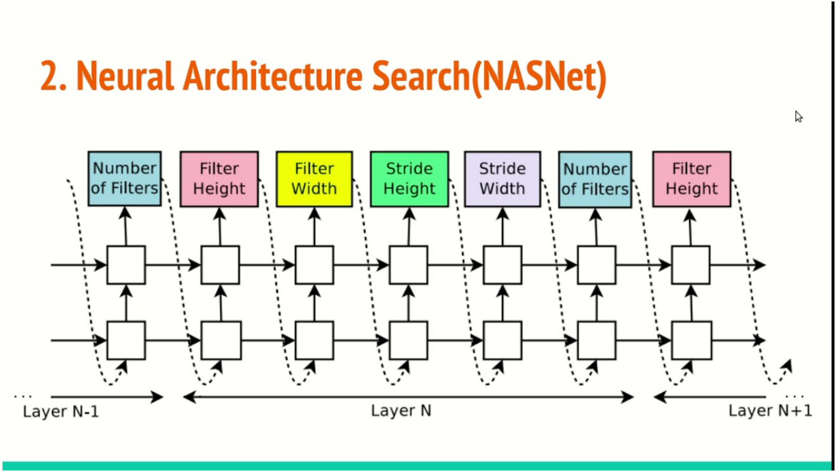
Inception V3 - Google (2015)



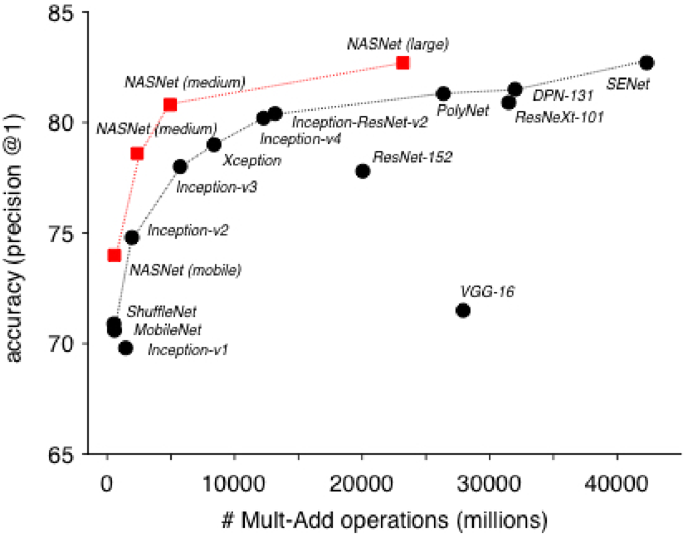
ResNet - Microsoft (2015)



NASNet - Google (2018)



Comparison



85
80
75
70
65
accuracy (precision @1)
0
10000
20000
30000
40000
Mult-Add operations (millions)
NASNet (large)
NASNet (medium)
NASNet (medium)
NASNet (mobile)
SENet
DPN-131
ResNeXt-101
PolyNet
Inception-ResNet-v2
Inception-v4
Xception
Inception-v3
ResNet-152
Inception-v2
VGG-16
ShuffleNet
MobileNet
Inception-v1

Keras pretrained Neural Network

```python
inception = InceptionV3(weights='imagenet', include_top=False)

y = inception.output
y = GlobalAveragePooling2D()(y)
y = Dense(1024, activation='relu')(y)
y = Dense(200, activation='relu')(y)
output = Dense(10, activation='softmax')(y)

model = Model(inputs=inception.input, outputs=output)
```