DEEP LEARNING

CSE 599 N1: Modern Mobile Systems

modernmobile.cs.washington.edu

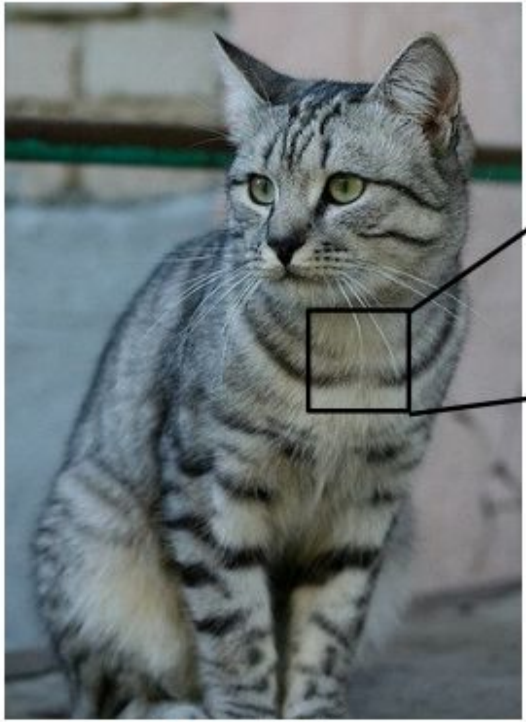
Content borrowed from Tianqi Chen, Fei-Fei Li

Image Classification: A core task in Computer Vision

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

→ cat

The Problem: Semantic Gap



This image by Nikita is licensed under CC-BY 2.0

```
[[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)

Distance Metric to compare images

L1 distance:

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

\-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

$\xrightarrow{\text{add}}$ 456

Parametric Approach: Linear Classifier

3072x1

$$\boxed{f(x,W)} = \boxed{W}\boxed{x} + \boxed{b}$$

10x1

10x1 10x3072

Image

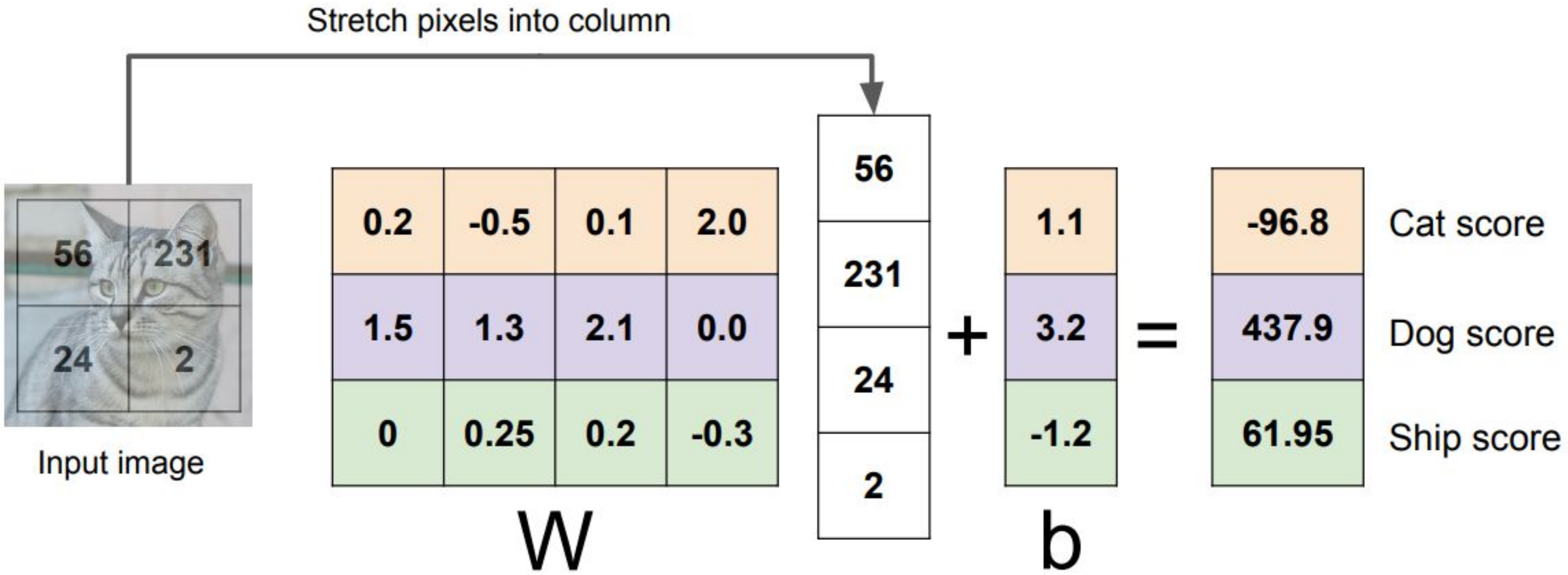
Array of **32x32x3** numbers
(3072 numbers total)

$f(\mathbf{x},\mathbf{W})$ →

10 numbers giving
class scores

W
parameters
or weights

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

A **loss function** tells how good our current classifier is

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Where x_i is image and y_i is (integer) label

Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

Loss over full dataset is average:

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

L = (2.9 + 0 + 12.9)/3
= **5.27**

Regularization

λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)} + \underbrace{\lambda R(W)}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Simple examples

L2 regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

More complex:

Dropout

Batch normalization

Stochastic depth, fractional pooling, etc

Regularization

λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)} + \underbrace{\lambda R(W)}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Why regularize?

- Express preferences over weights
- Make the model *simple* so it works on test data
- Improve optimization by adding curvature

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$ Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$

	Unnormalized log-probabilities / logits		unnormalized probabilities		probabilities		Correct probs
cat	**3.2**	exp →	**24.5**	normalize →	**0.13**	→ compare ←	**1.00**
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Neural networks: without the brain stuff

(**Before**) Linear score function: $f = Wx$

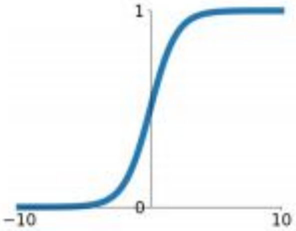
(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$
or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

Activation functions

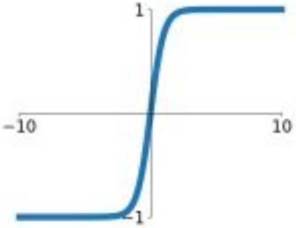
Sigmoid

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



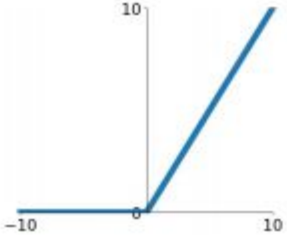
tanh

$\tanh(x)$



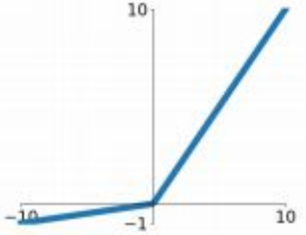
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

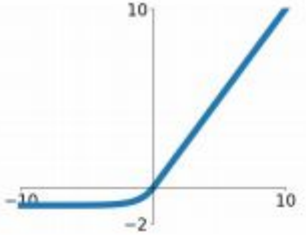


Maxout

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

ELU

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



Elements of Machine Learning

Model

$$x_i = \begin{bmatrix} \text{feature}_0 \\ \text{feature}_1 \\ \cdots \\ \text{feaure}_m \end{bmatrix} \qquad \hat{y}_i = \frac{1}{1 + \exp\left(-w^T x_i\right)}$$

Objective

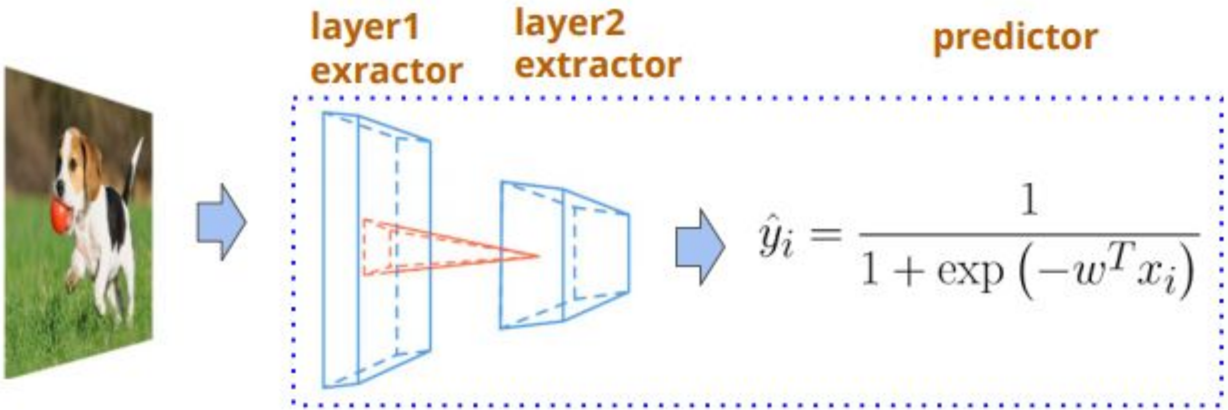
$$L(w) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \lambda \|w\|^2$$

Training

$$w \leftarrow w - \eta \nabla_w L(w)$$

What's Special About Deep Learning

Compositional Model



$$\hat{y}_i = \frac{1}{1+\exp\left(-w^T x_i\right)}$$

End to End Training

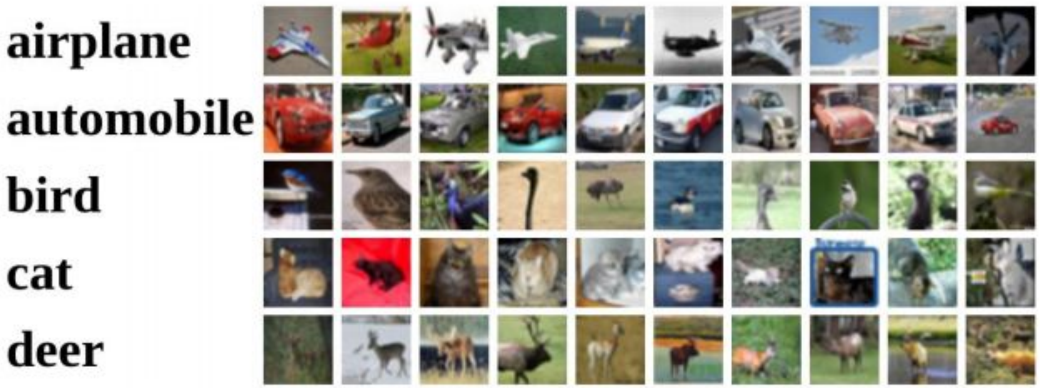
Machine Learning: Data-Driven Approach

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
```

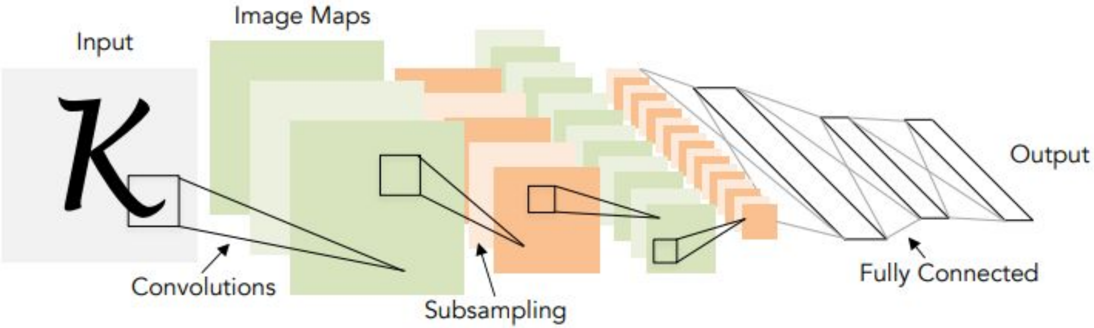
```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

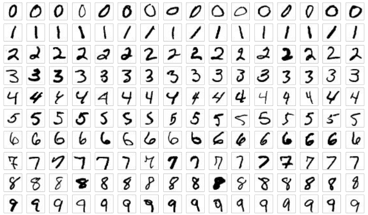
Example training set



1998

LeCun et al.





# of transistors

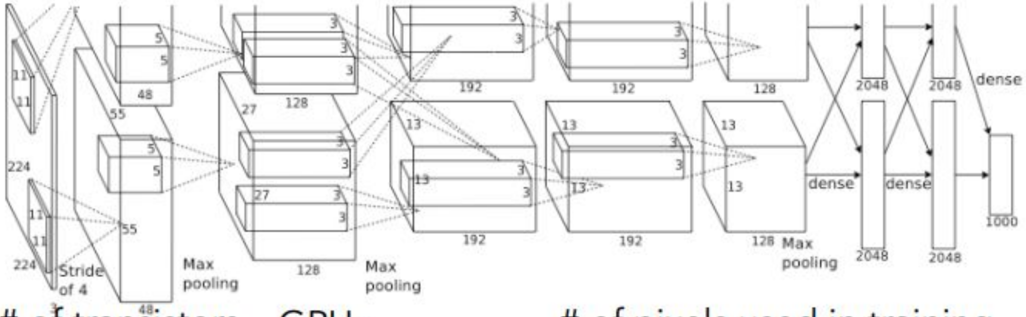
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



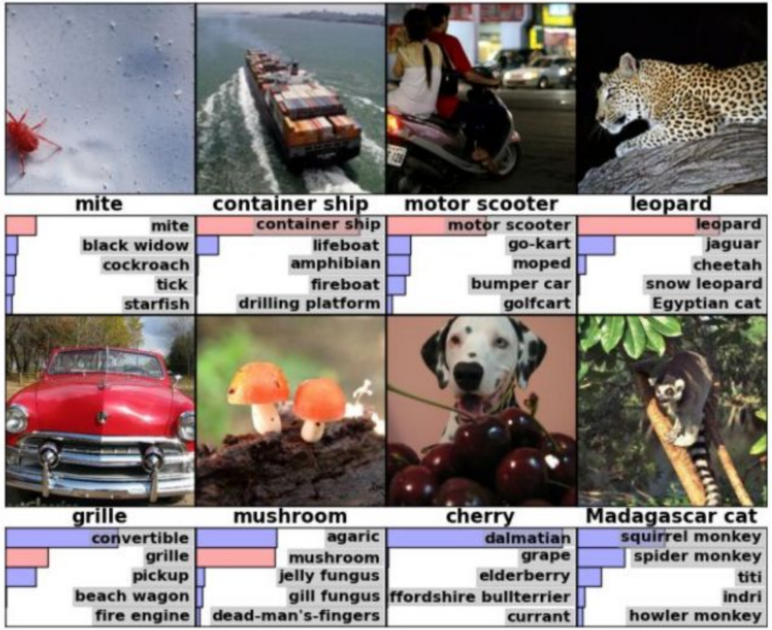
# of transistors

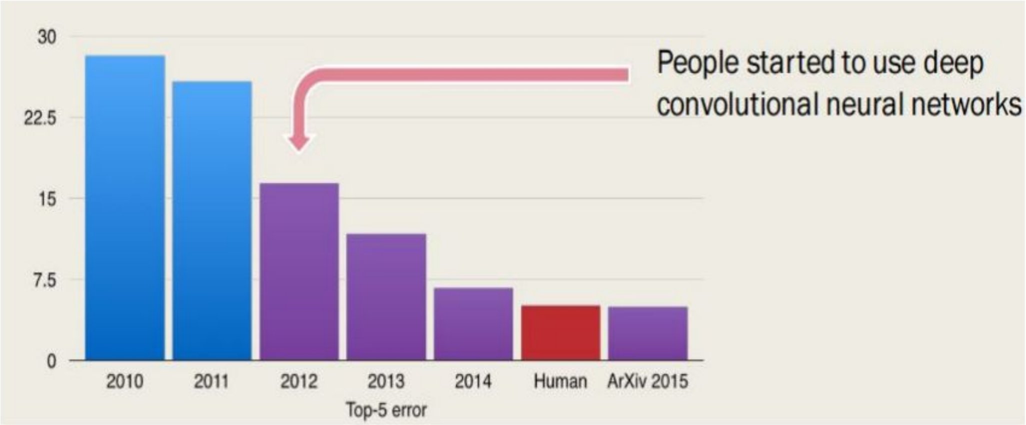
10^9

GPUs

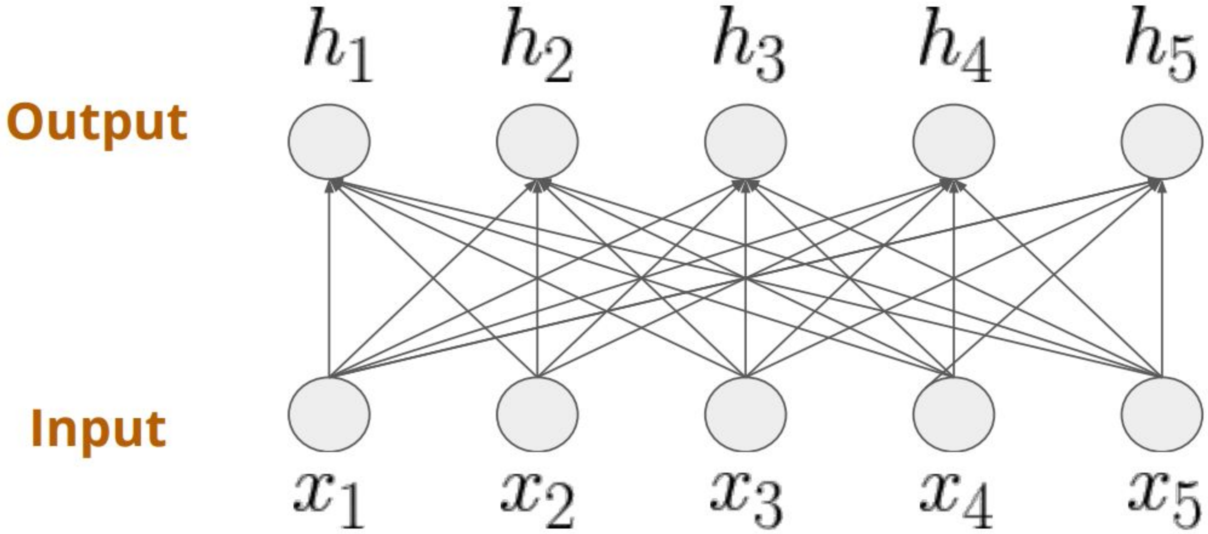
# of pixels used in training

10^{14} IMAGENET


mite
container ship
motor scooter
leopard
mite
black widow
cockroach
tick
starfish
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
go-kart
moped
bumper car
golfcart
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
mushroom
cherry
Madagascar cat
convertible
grille
pickup
beach wagon
fire engine
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
dalmatian
grape
elderberry
ffordshire bullterrier
currant
squirrel monkey
spider monkey
titi
indri
howler monkey

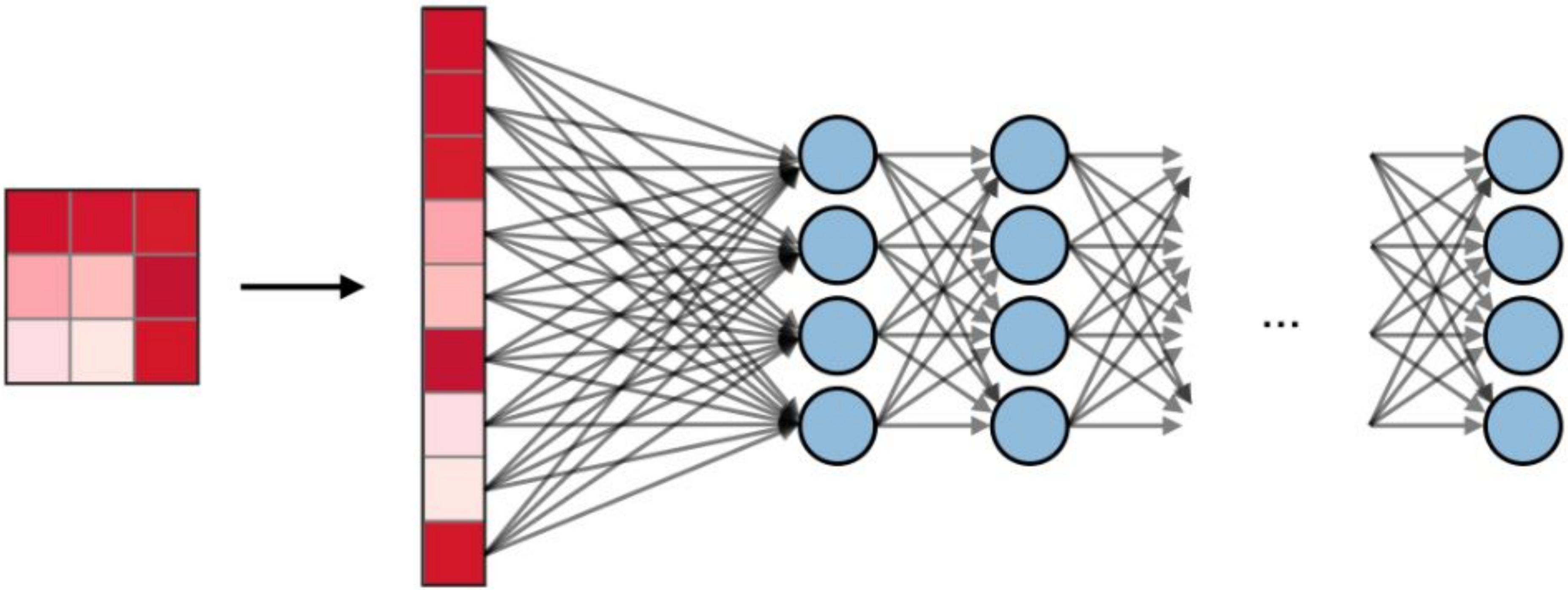

30
22.5
15
7.5
0
People started to use deep convolutional neural networks
2010
2011
2012
2013
2014
Human
ArXiv 2015
Top-5 error

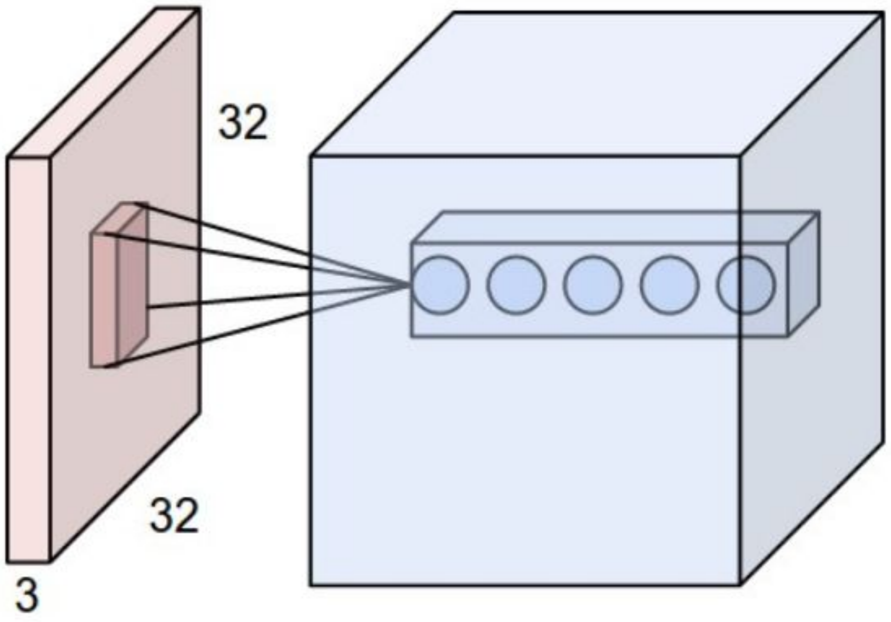
Fully Connected Layer

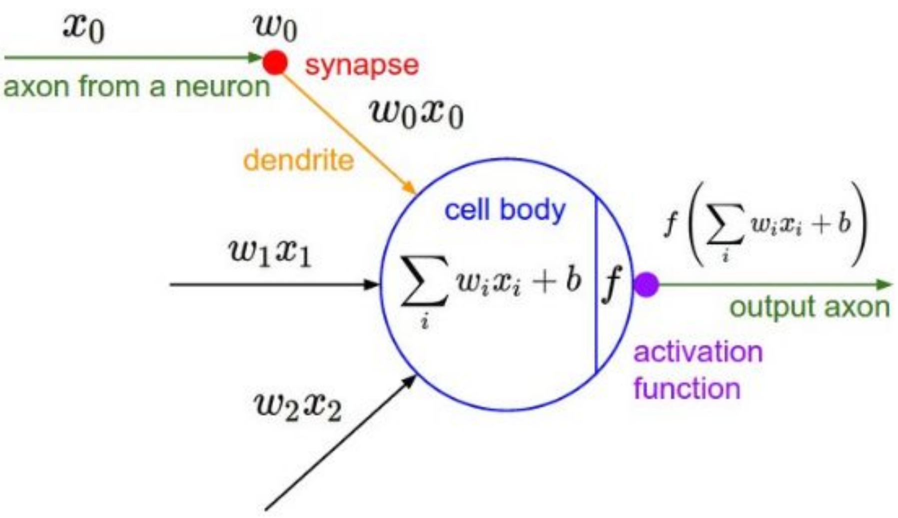


$$h_i = \sum_{j=1}^{5} W_{ij} x_i$$

$$h_1 = W_{11}x_1 + W_{21}x_2 + W_{31}x_3 + W_{41}x_4 + W_{51}x_5$$

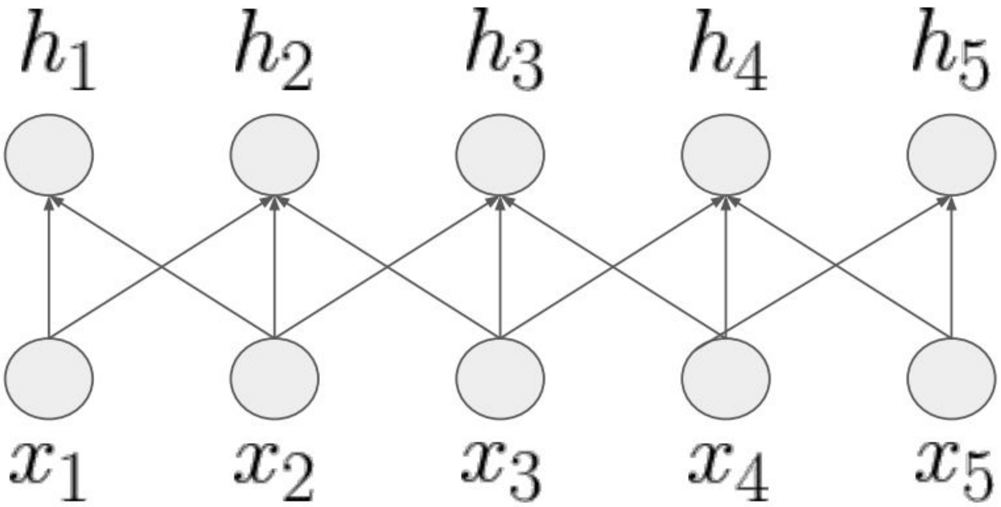



32
32
3


x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$f\left(\sum_i w_i x_i + b\right)$
$w_1 x_1$
$\sum_i w_i x_i + b$
f
output axon
activation
function
$w_2 x_2$

Convolution = Spatial Locality + Sharing

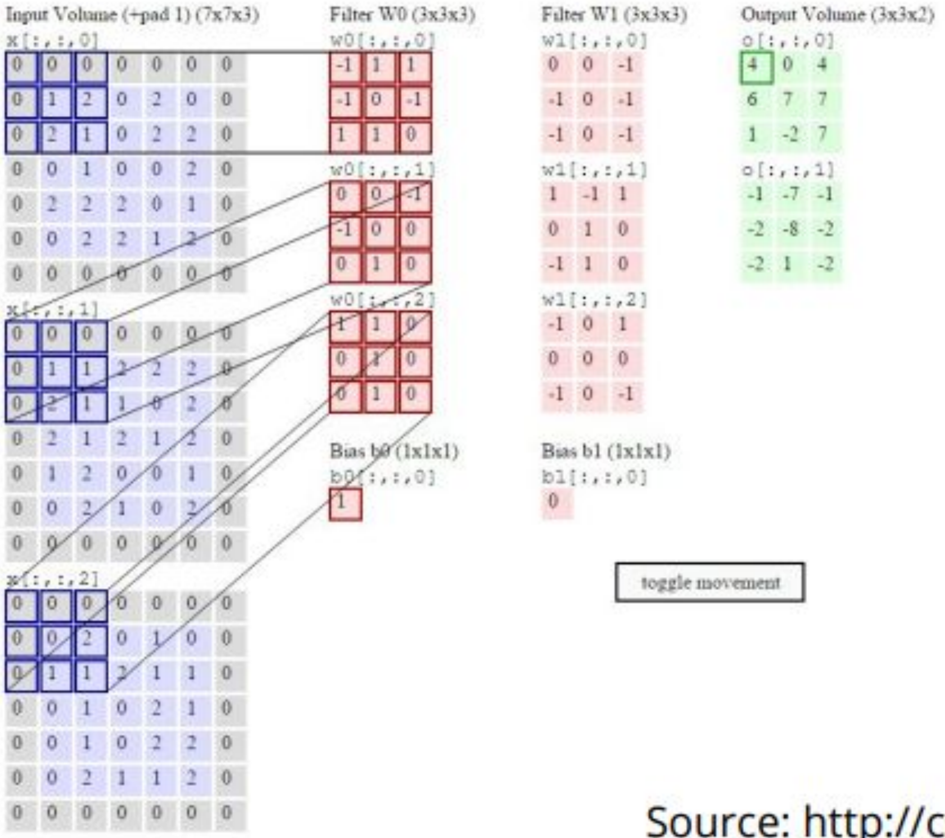
Spatial Locality

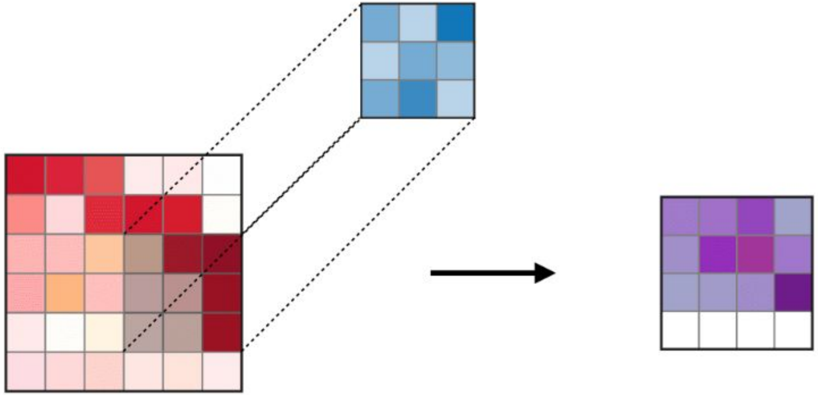


Without Sharing $h_i = W_{1,i}x_{i-1} + W_{2,i}x_i + W_{3,i}x_{i+1}$

With Sharing $h_i = W_1 x_{i-1} + W_2 x_i + W_3 x_{i+1}$

Convolution with Multiple Channels

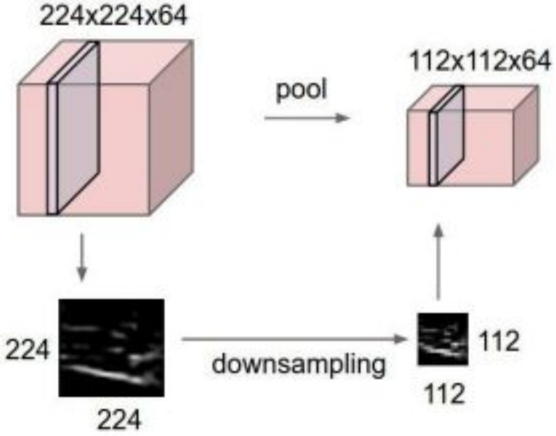


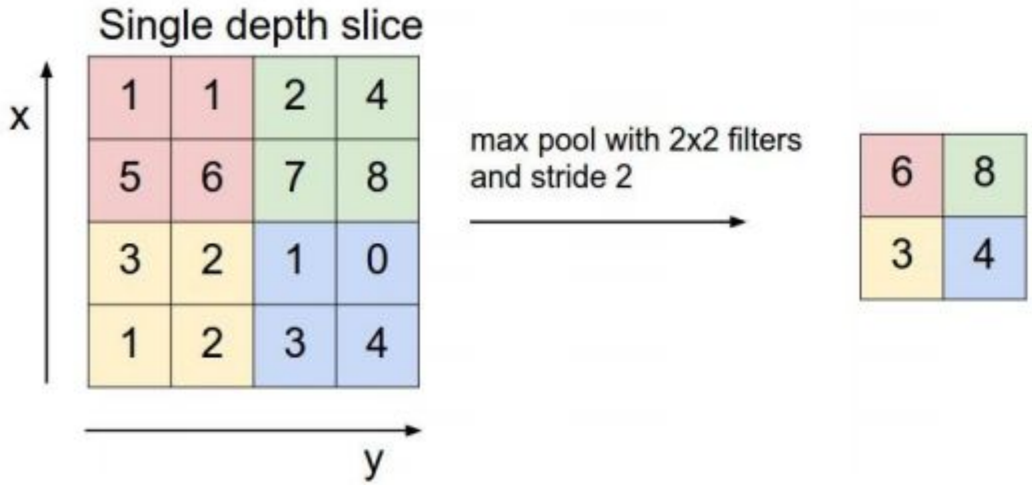


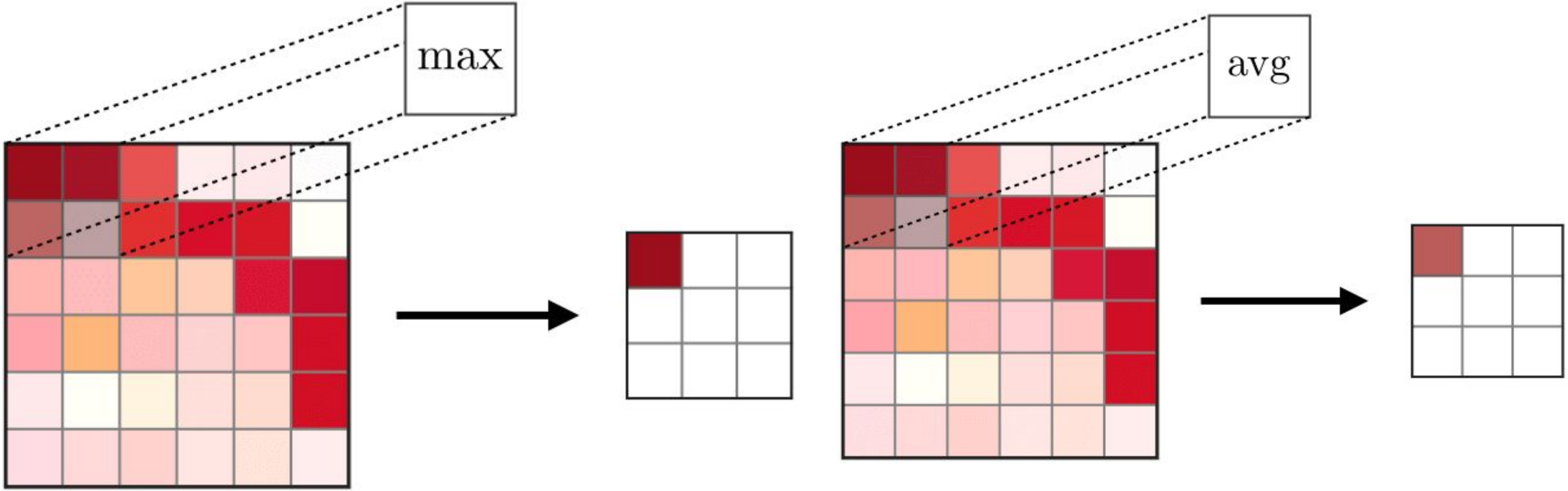
Source: http://cs231n.github.io/convolutional-networks/

Pooling Layer

Can be replaced by strided convolution






max
avg

Challenges: From LeNet to AlexNet

1.2 million images with 1000 categories

- Need much more data: ImageNet
- A lot more computation burdens: GPU

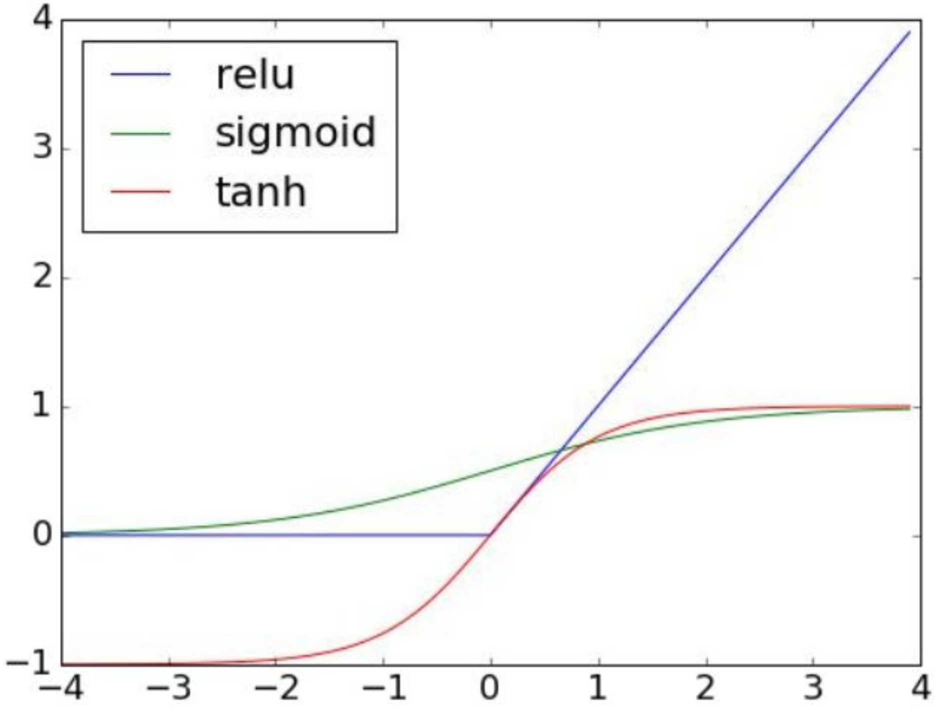
- Overfitting prevention
  - Dropout regularization

- Stable initialization and training
  - Explosive/vanishing gradient problems
  - Requires careful tuning of initialization and data normalization

ReLU Unit

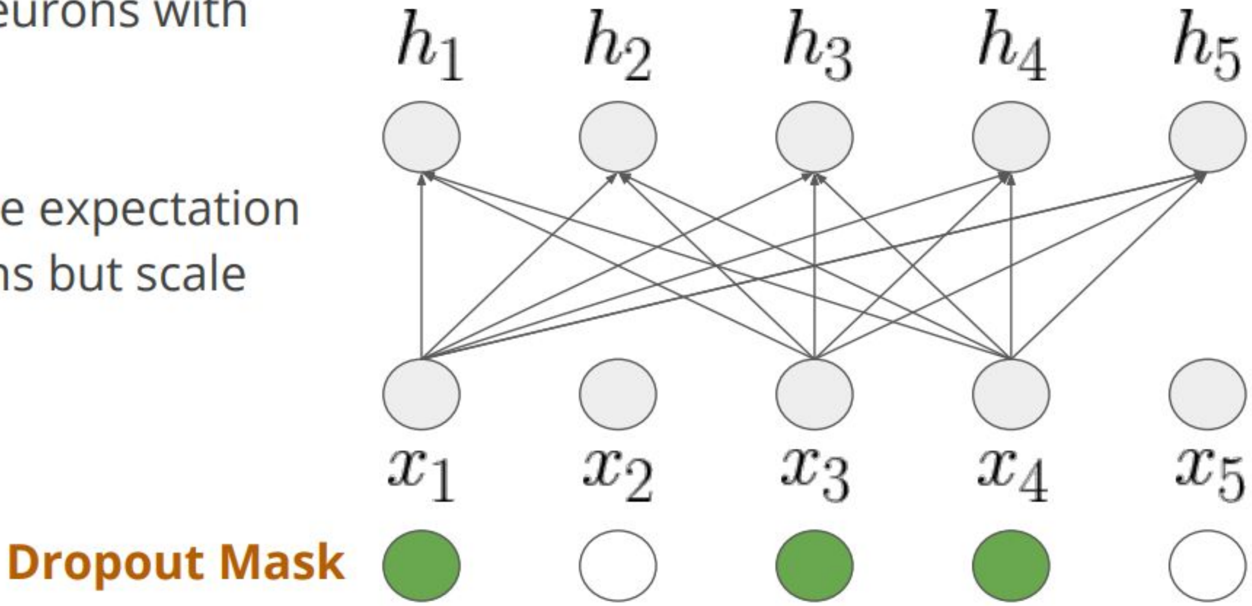
- ReLU $y = max(x, 0)$

- Why ReLU?
  - Cheap to compute
  - It is roughly linear..



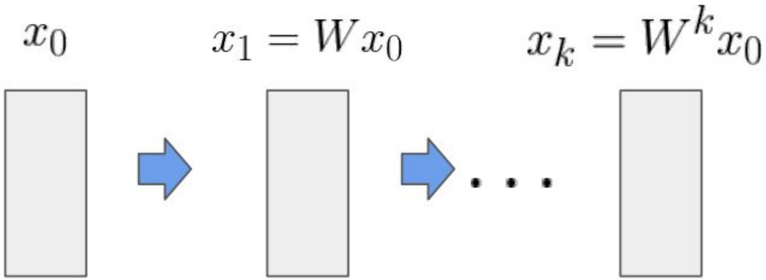
Dropout Regularization

- Randomly zero out neurons with probability 0.5
- During prediction, use expectation value (keep all neurons but scale output by 0.5)



Vanishing and Explosive Value Problem

- Imagine each layer multiplies
  Its input by same weight matrix
  - W > 1: exponential explosion
  - W < 1: exponential vanishing



- In ConvNets, the weight are not tied, but their magnitude matters
  - Deep nets training **was** initialization sensitive

Batch Normalization: Stabilize the Magnitude

- Subtract mean
- Divide by standard deviation
- Output is invariant to input scale!
  - Scale input by a constant
  - Output of BN remains the same

- Impact
  - Easy to tune learning rate
  - Less sensitive initialization

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

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

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

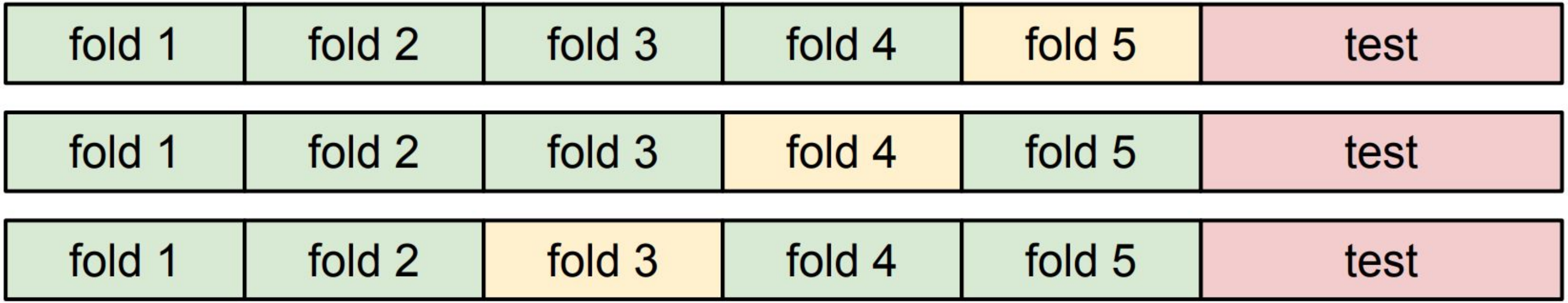
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

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

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Setting Hyperparameters

Idea #4: **Cross-Validation**: Split data into **folds**, try each fold as validation and average the results



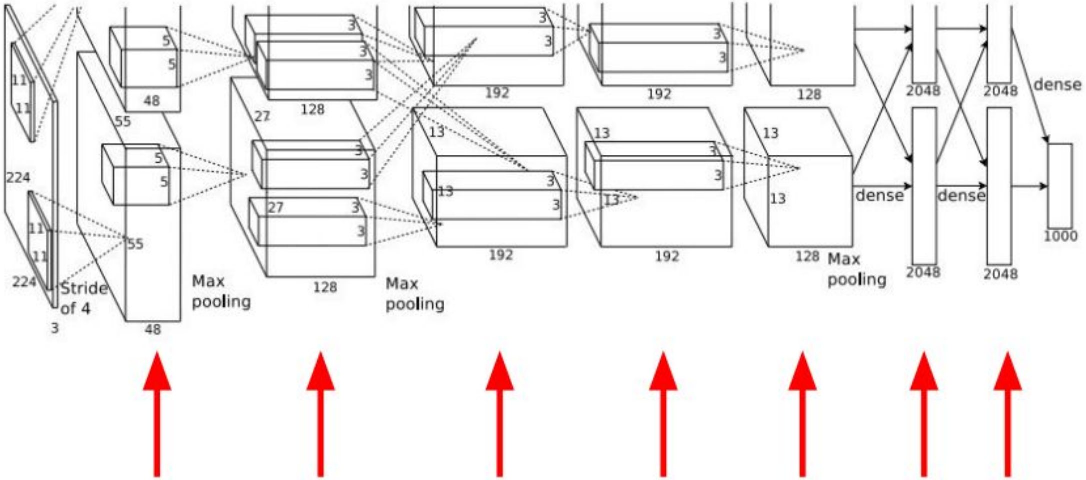
Useful for small datasets, but not used too frequently in deep learning

Understanding what a neural network has learned

What’s going on inside ConvNets?

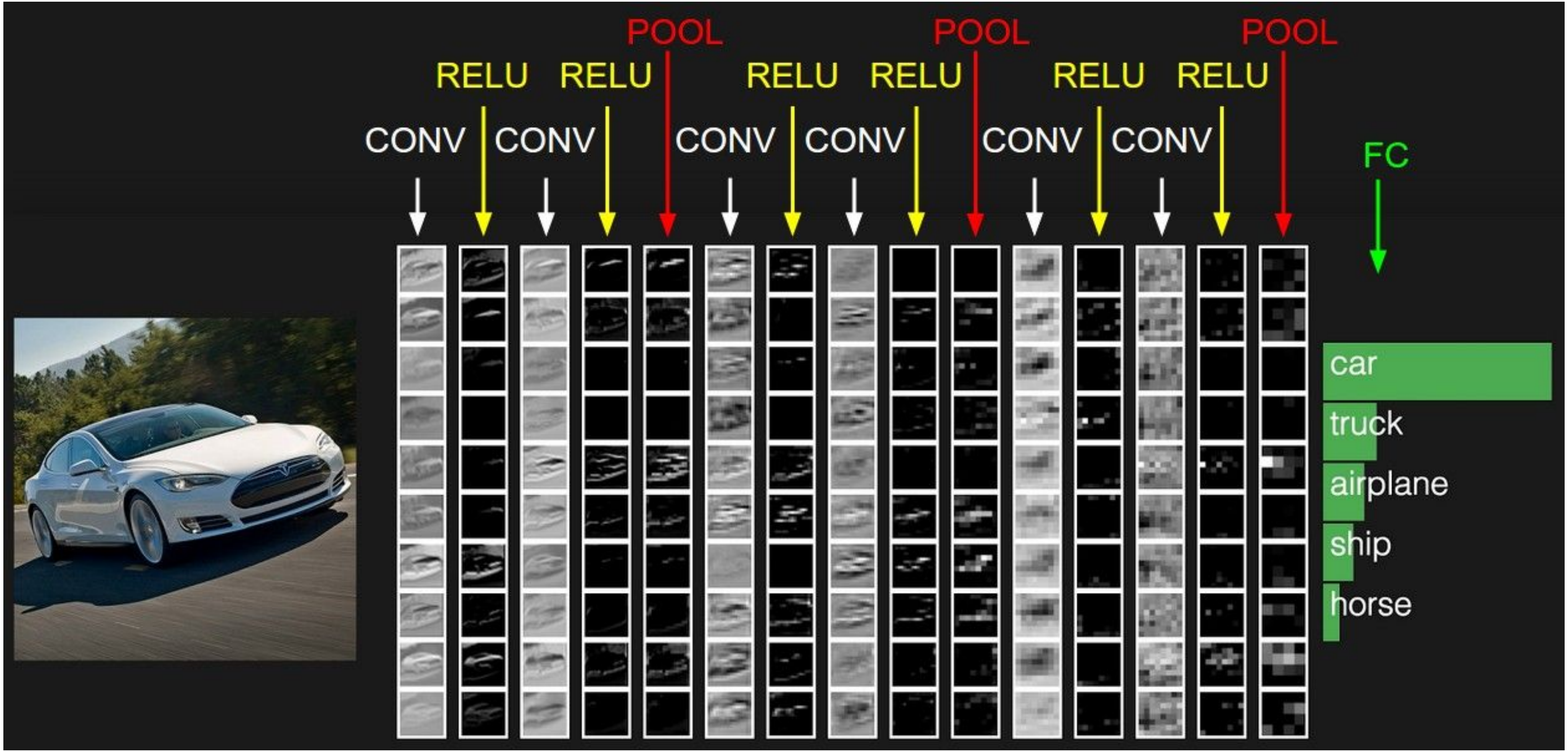
This image is CC0 public domain

Input Image:
3 x 224 x 224

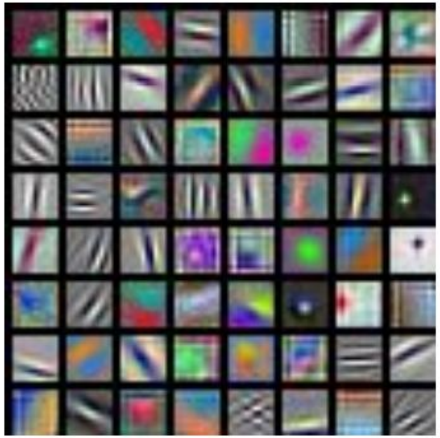


Class Scores:
1000 numbers

What are the intermediate features looking for?


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

First Layer: Visualize Filters

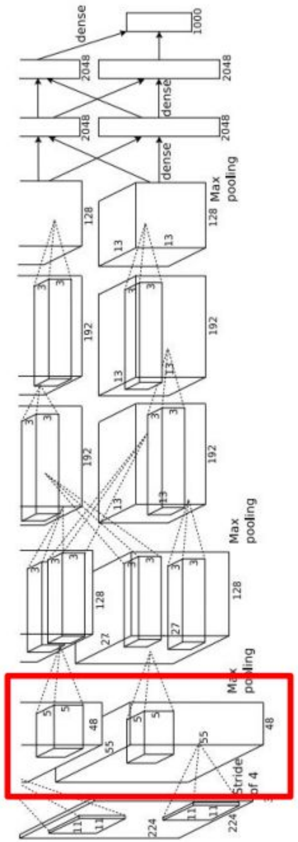


AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7



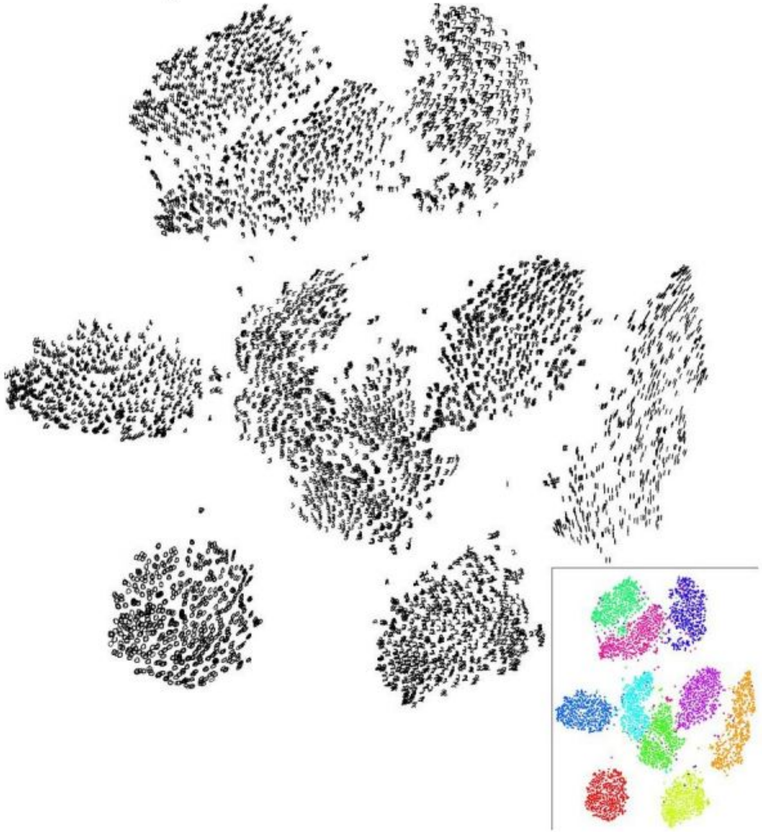
Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

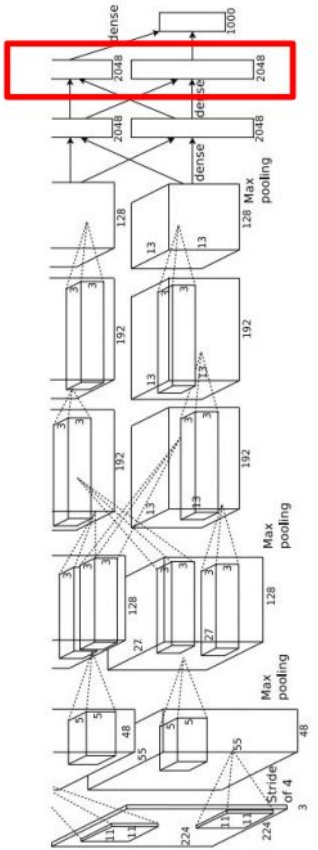
Last Layer: Dimensionality Reduction

Visualize the “space” of FC7 feature vectors by reducing dimensionality of vectors from 4096 to 2 dimensions

Simple algorithm: Principal Component Analysis (PCA)

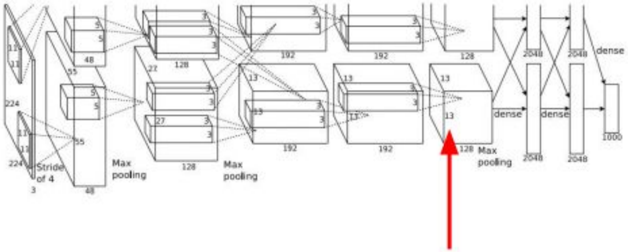
More complex: **t-SNE**





Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008

Maximally Activating Patches



Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick channel 17/128

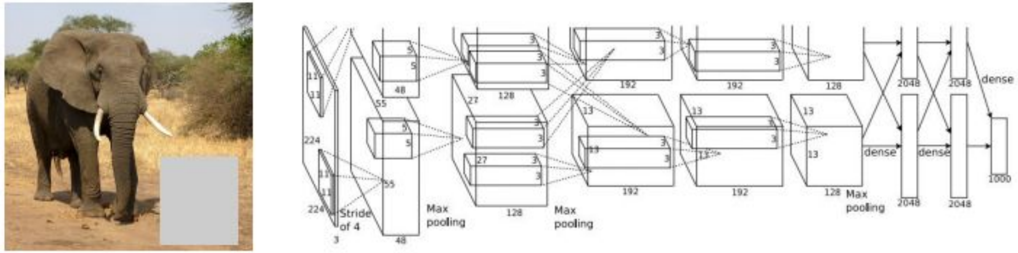
Run many images through the network, record values of chosen channel

Visualize image patches that correspond to maximal activations

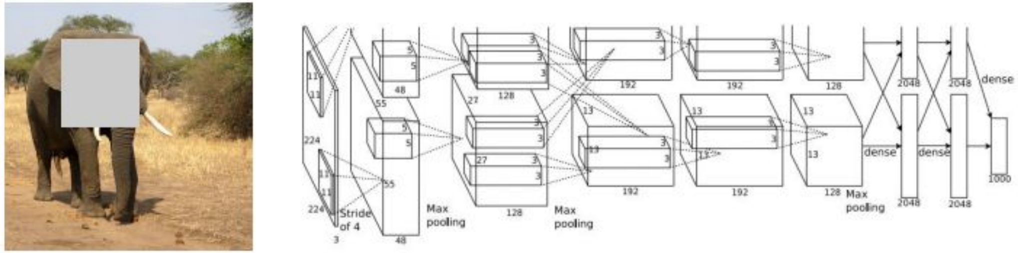
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

Which pixels matter: Saliency vs Occlusion

Mask part of the image before feeding to CNN, check how much predicted probabilities change



P(elephant) = 0.95

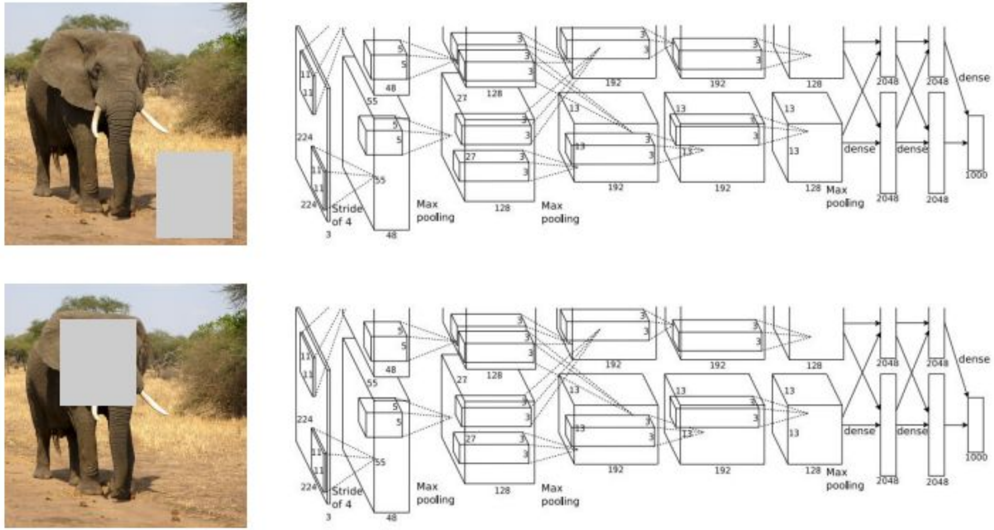


P(elephant) = 0.75

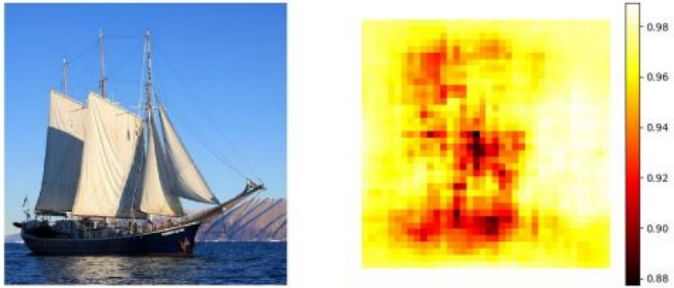
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Which pixels matter: Saliency vs Occlusion

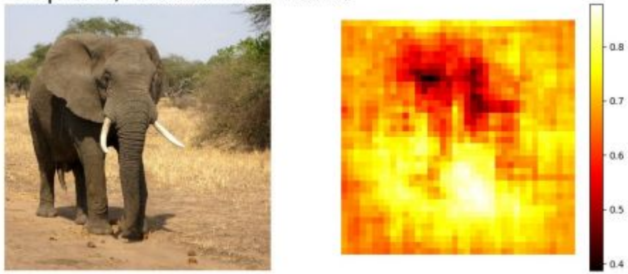
Mask part of the image before feeding to CNN, check how much predicted probabilities change



schooner



African elephant, Loxodonta africana



go-kart

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Transfer learning

Transfer learning (fixed feature extractor)

Too costly to train your own CNN (2-3 weeks on GPUs)

Use a CNN pretrained on ImageNet and adapt it to your own dataset

Final layer of CNN is a ‘dense’ layer with # of nodes == # of classes (1000 for ImageNet)

Remove final layer, replace with Dense layer with your # of nodes, with a softmax activation

Transfer learning (fine-tuning)

Fine-tune the weights of the n-1th convolution layer

Earlier layers encode most abstract features, lines, edges etc.

Later layers are biased towards correctly classifying the ImageNet dataset

Model Zoo

https://github.com/BVLC/caffe/wiki/Model-Zoo

Model Zoo

Aswin Shanmugam Subramanian edited this page 4 days ago · 120 revisions

Check out the model zoo documentation for details.

To acquire a model:

1. download the model gist by `./scripts/download_model_from_gist.sh <gist_id> <dirname>` to load the model metadata, architecture, solver configuration, and so on. (`<dirname>` is optional and defaults to caffe/models).
2. download the model weights by `./scripts/download_model_binary.py <model_dir>` where `<model_dir>` is the gist directory from the first step.

or visit the [model zoo documentation] (http://caffe.berkeleyvision.org/model_zoo.html) for complete instructions.

Table of Contents

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	99 MB	0.749	0.921	25,636,712	168
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

Health applications

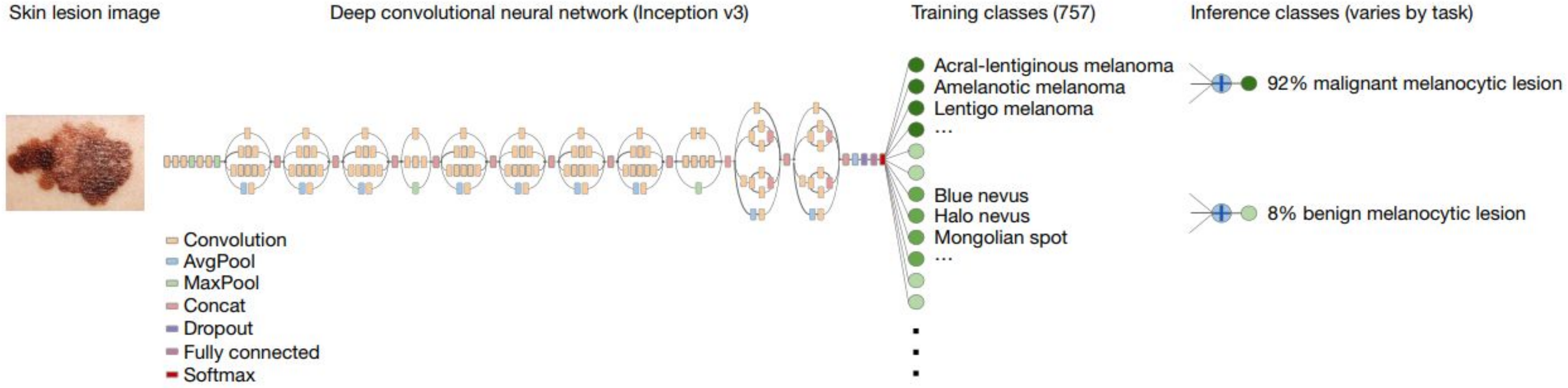
Dermatologist-level classification of skin cancer with deep neural ...

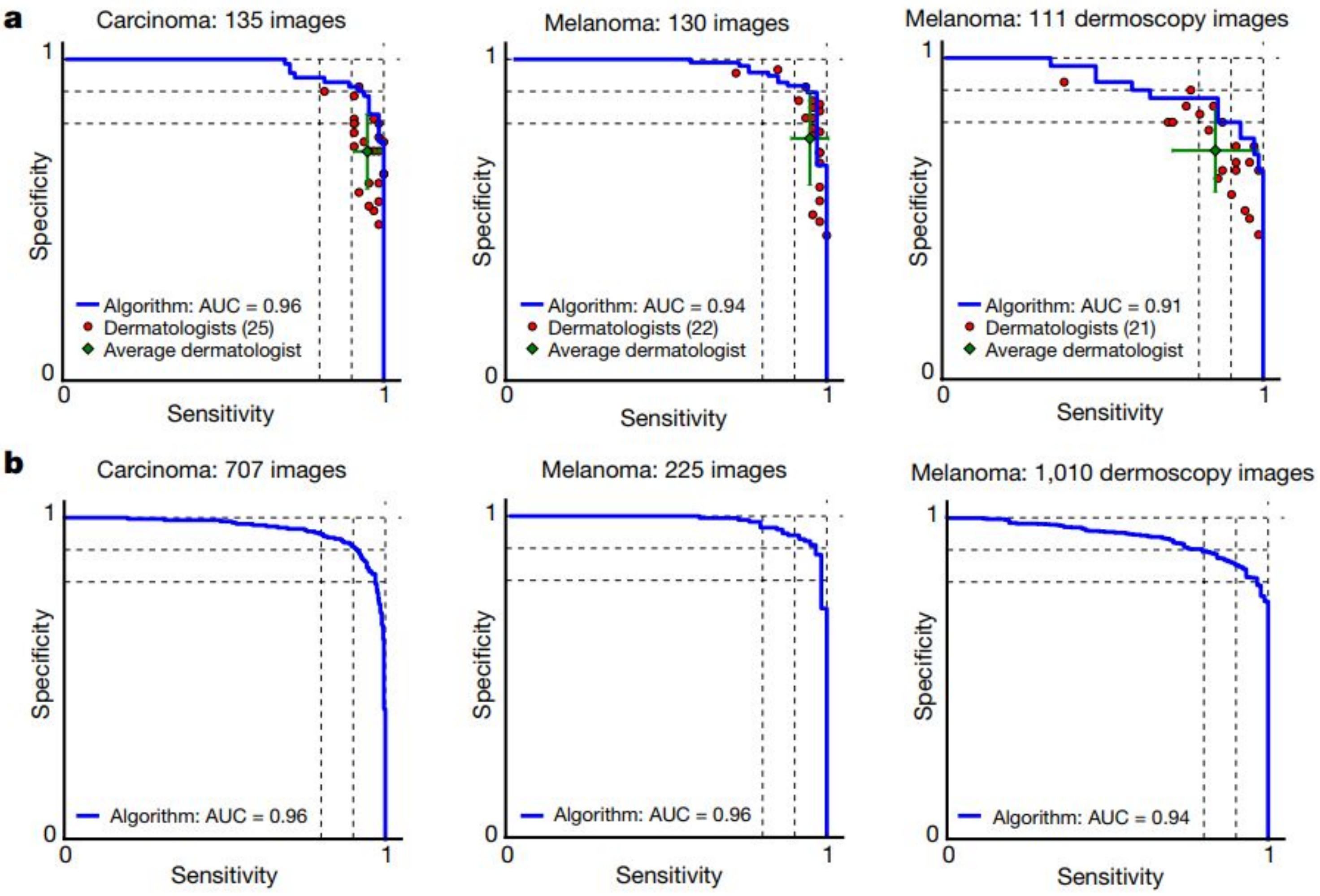
https://www.nature.com › letters

by A Esteva - 2017 - Cited by 1211 - Related articles

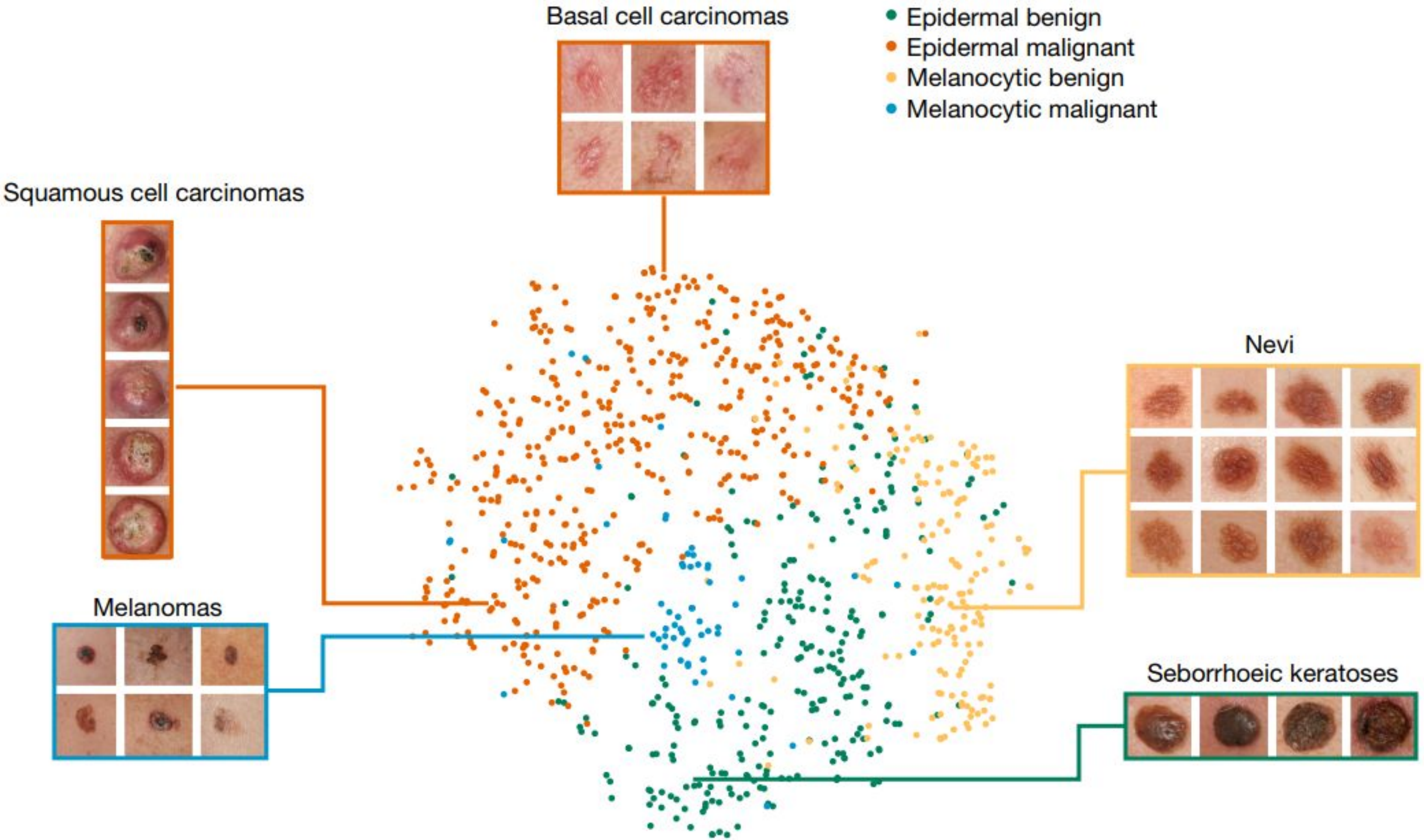
Jan 25, 2017 - a, The **deep learning** CNN outperforms the average of the **dermatologists** at **skin cancer classification** using photographic and dermoscopic images. Our CNN is tested against at least 21 **dermatologists** at keratinocyte carcinoma and **melanoma** recognition. ... The CNN outputs a malignancy probability P per image.

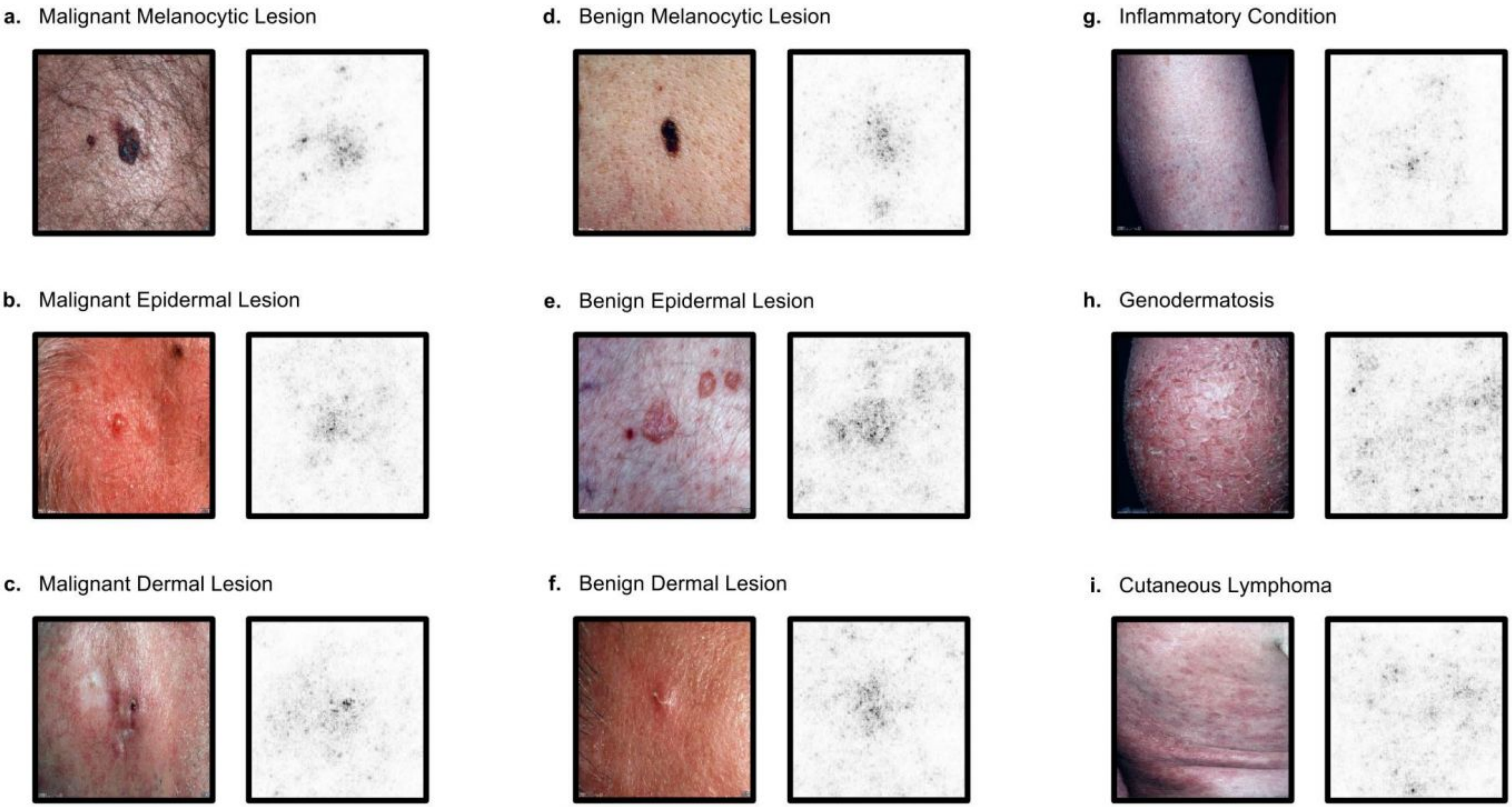
You've visited this page 3 times. Last visit: 11/18/18

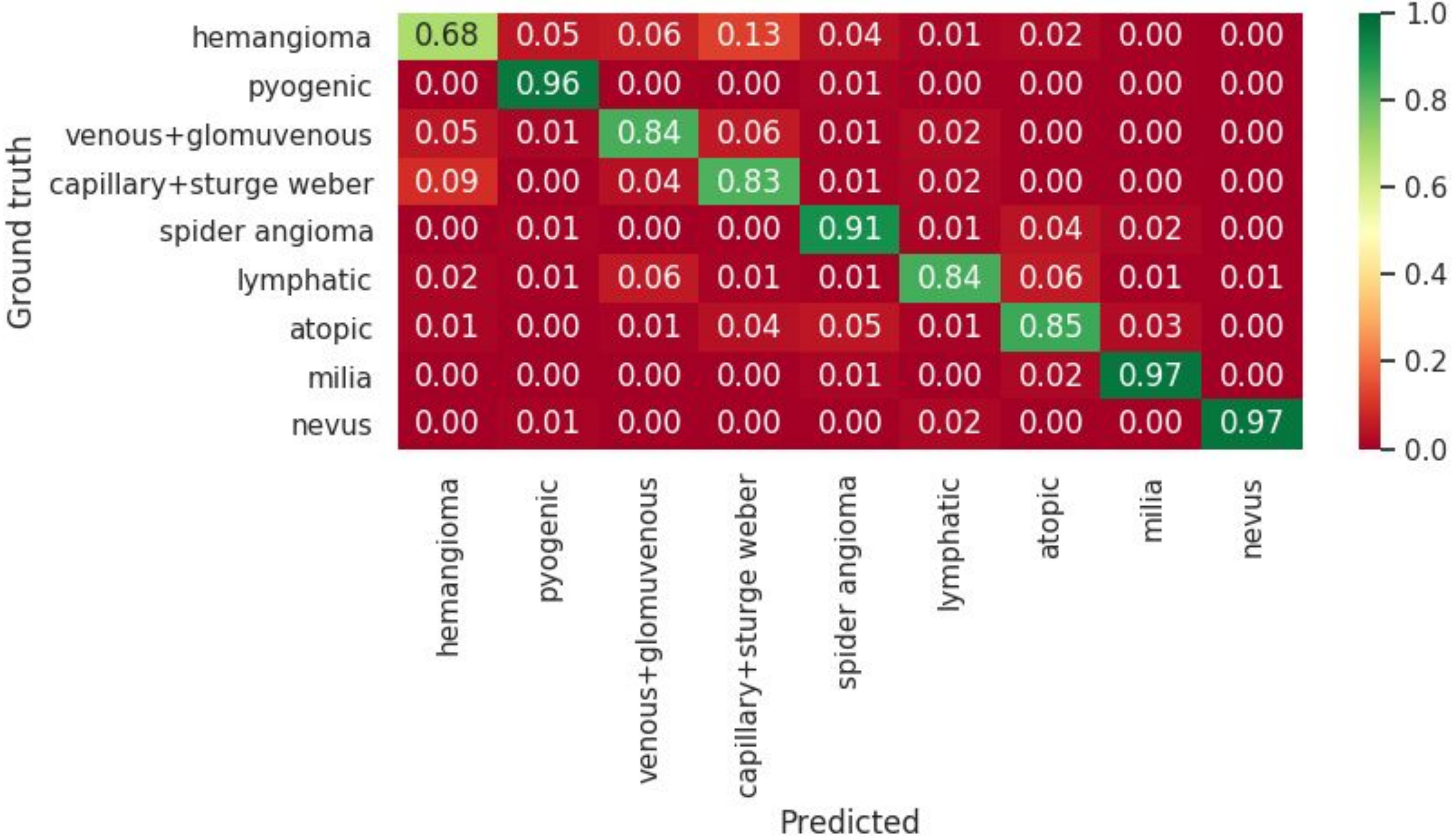


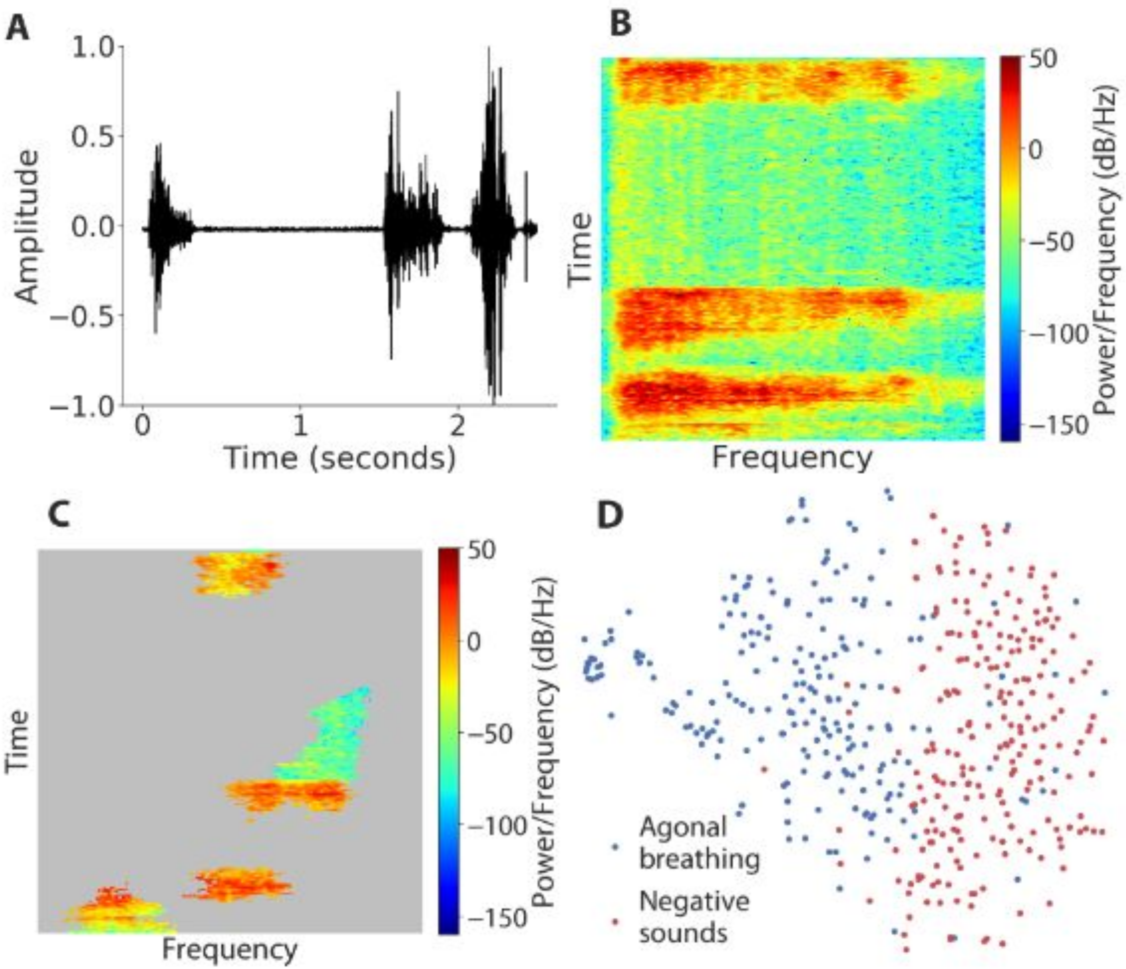


a
Carcinoma: 135 images
Melanoma: 130 images
Melanoma: 111 dermoscopy images
Specificity
Sensitivity
Algorithm: AUC = 0.96
Dermatologists (25)
Average dermatologist
Algorithm: AUC = 0.94
Dermatologists (22)
Average dermatologist
Algorithm: AUC = 0.91
Dermatologists (21)
Average dermatologist
b
Carcinoma: 707 images
Melanoma: 225 images
Melanoma: 1,010 dermoscopy images
Algorithm: AUC = 0.96
Algorithm: AUC = 0.96
Algorithm: AUC = 0.94

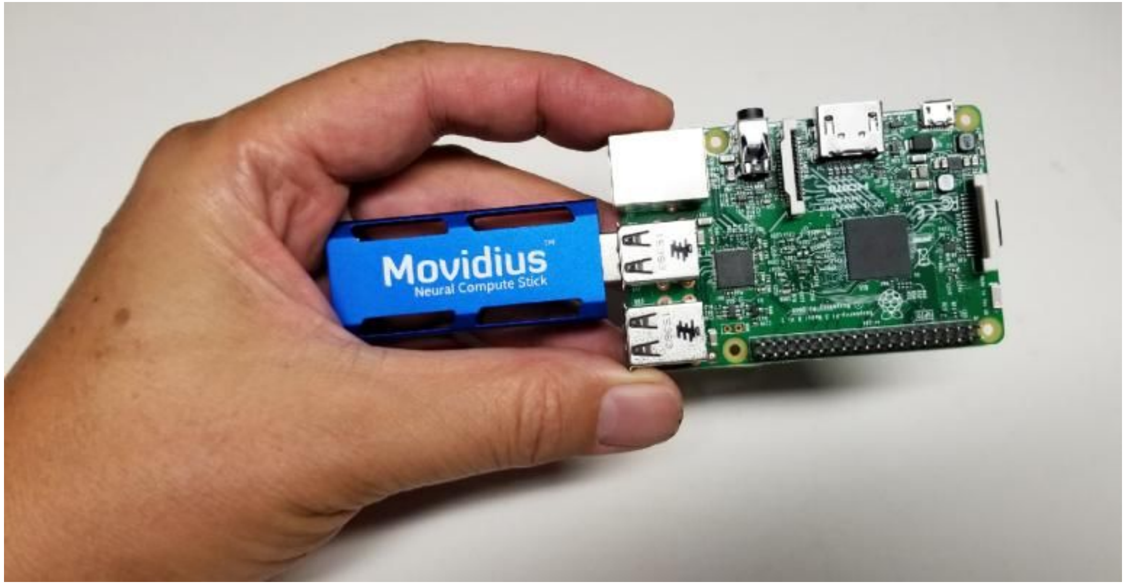

Basal cell carcinomas
Epidermal benign
Epidermal malignant
Melanocytic benign
Melanocytic malignant
Squamous cell carcinomas
Nevi
Melanomas
Seborrhoeic keratoses


a. Malignant Melanocytic Lesion
b. Malignant Epidermal Lesion
c. Malignant Dermal Lesion
d. Benign Melanocytic Lesion
e. Benign Epidermal Lesion
f. Benign Dermal Lesion
g. Inflammatory Condition
h. Genodermatosis
i. Cutaneous Lymphoma


hemangioma
pyogenic
venous+glomuvenous
capillary+sturge weber
spider angioma
lymphatic
atopic
milia
nevus
Ground truth
Predicted
0.68 0.05 0.06 0.13 0.04 0.01 0.02 0.00 0.00
0.00 0.96 0.00 0.00 0.01 0.00 0.00 0.00 0.00
0.05 0.01 0.84 0.06 0.01 0.02 0.00 0.00 0.00
0.09 0.00 0.04 0.83 0.01 0.02 0.00 0.00 0.00
0.00 0.01 0.00 0.00 0.91 0.01 0.04 0.02 0.00
0.02 0.01 0.06 0.01 0.01 0.84 0.06 0.01 0.01
0.01 0.00 0.01 0.04 0.05 0.01 0.85 0.03 0.00
0.00 0.00 0.00 0.00 0.01 0.00 0.02 0.97 0.00
0.00 0.01 0.00 0.00 0.00 0.02 0.00 0.00 0.97
1.0
0.8
0.6
0.4
0.2
0.0


A
1.0
0.5
0.0
−0.5
−1.0
Amplitude
0
1
2
Time (seconds)
B
Time
Frequency
50
0
−50
−100
−150
Power/Frequency (dB/Hz)
C
Time
Frequency
50
0
−50
−100
−150
Power/Frequency (dB/Hz)
D
Agonal breathing
Negative sounds

Deep learning for wireless networks


Movidius
Neural Compute Stick


RTLSDR

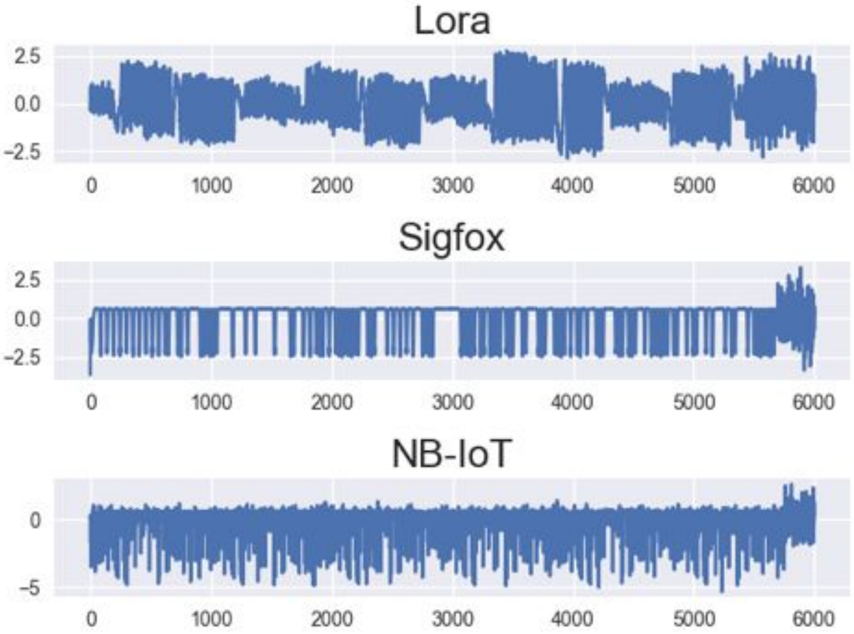
Lora, Sigfox, NB-IoT

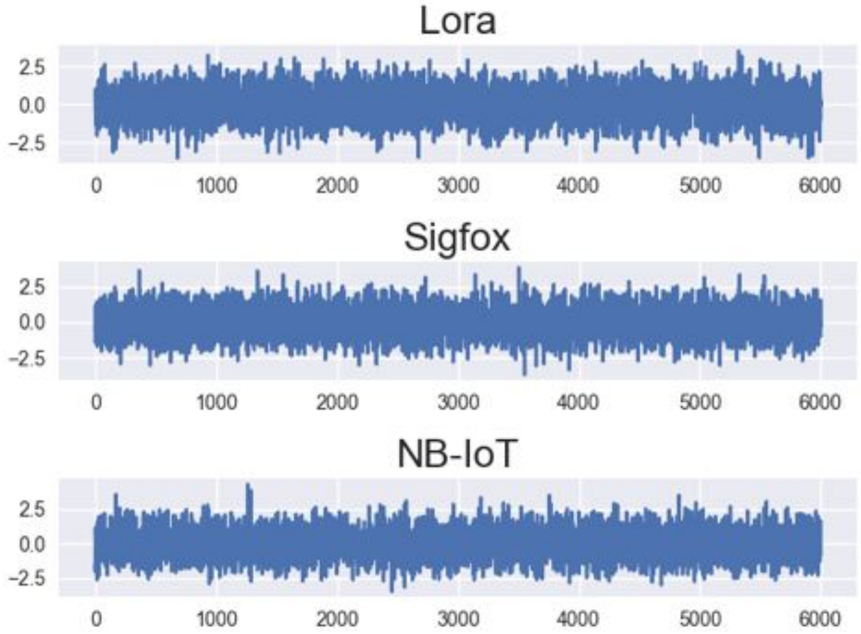
Long range + can be decoded BELOW the noise floor

Lora range: > 10 miles

No MAC protocols yet!

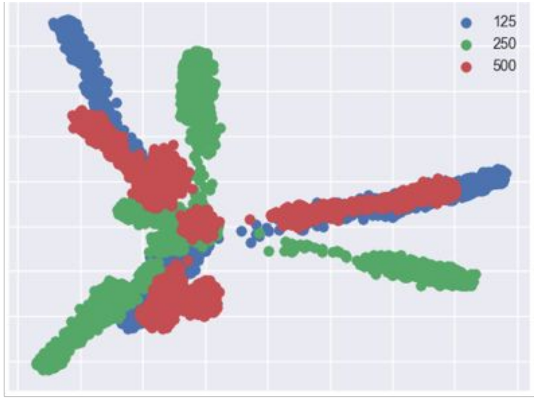
Lora, Sigfox and Z-wave are proprietary (unlike Wi-Fi)



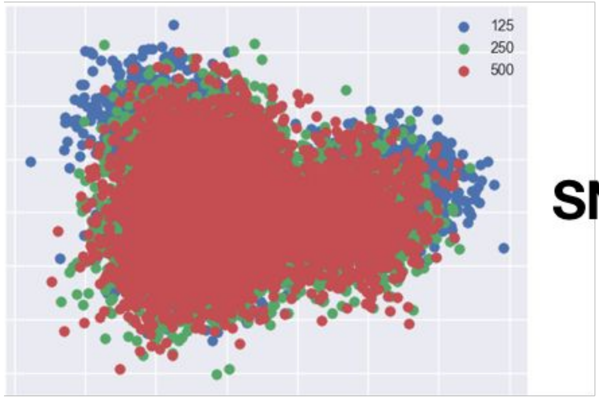


Dimensionality reduction

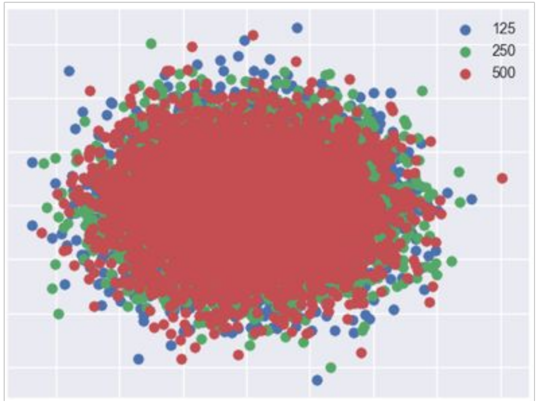
SNR=-10



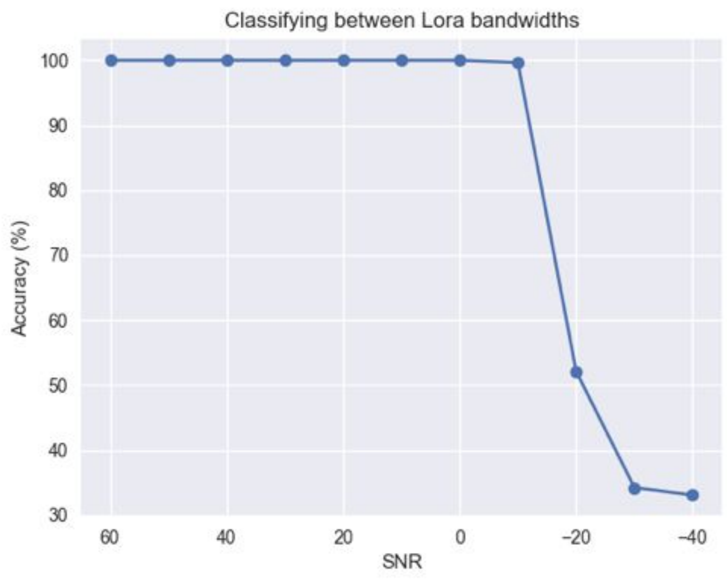
SNR=-20

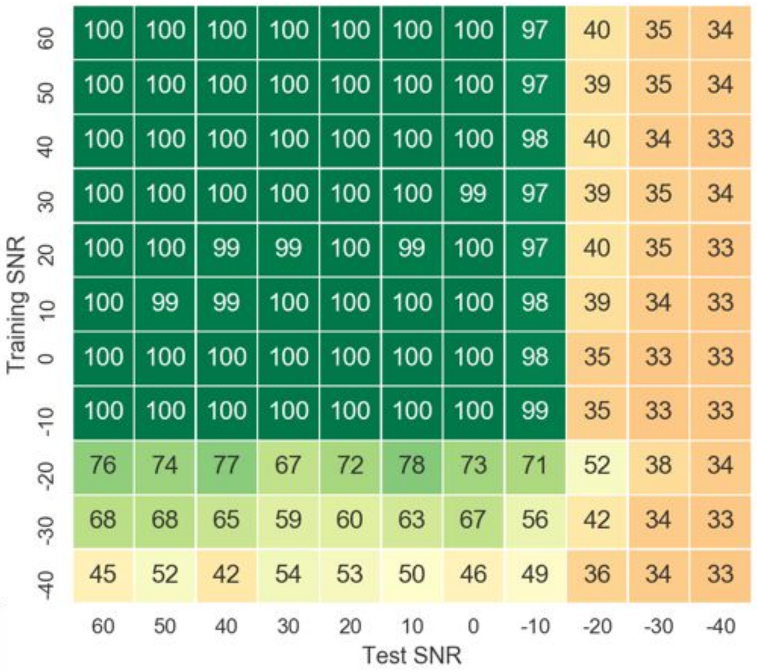


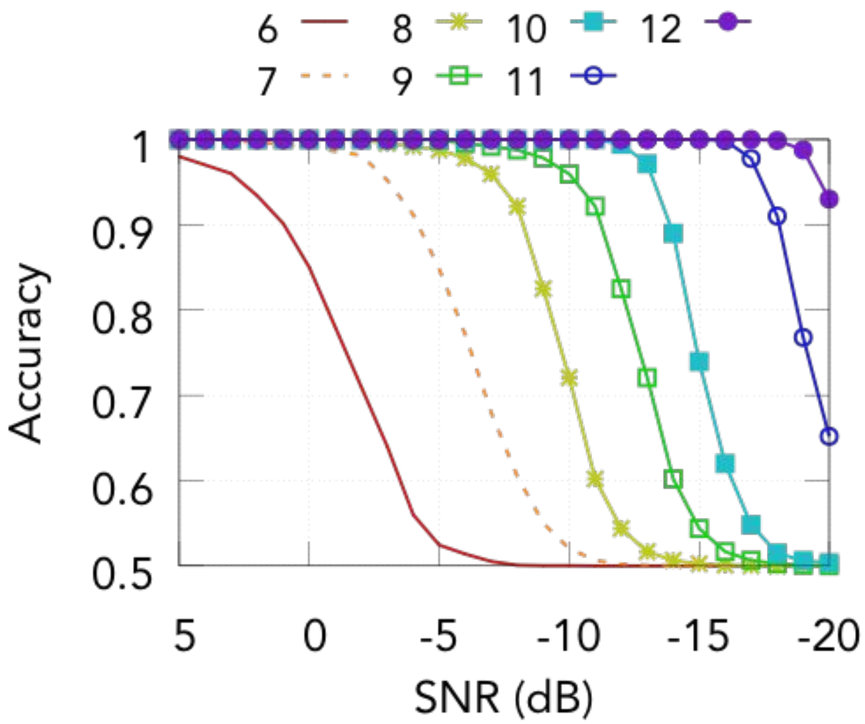
SNR=-30



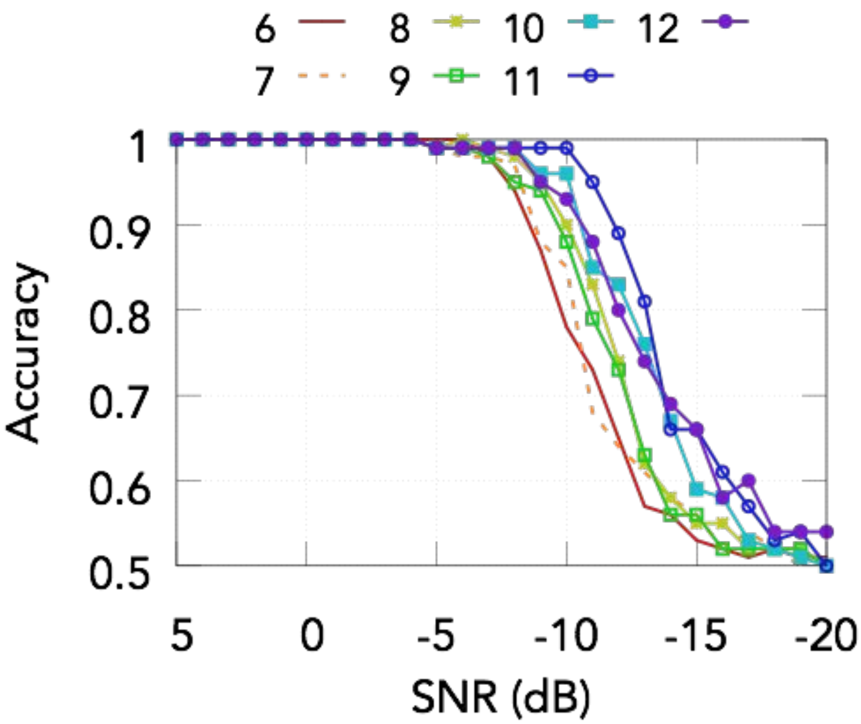
Machine learning underneath the noise floor







Baseline



DeepSense

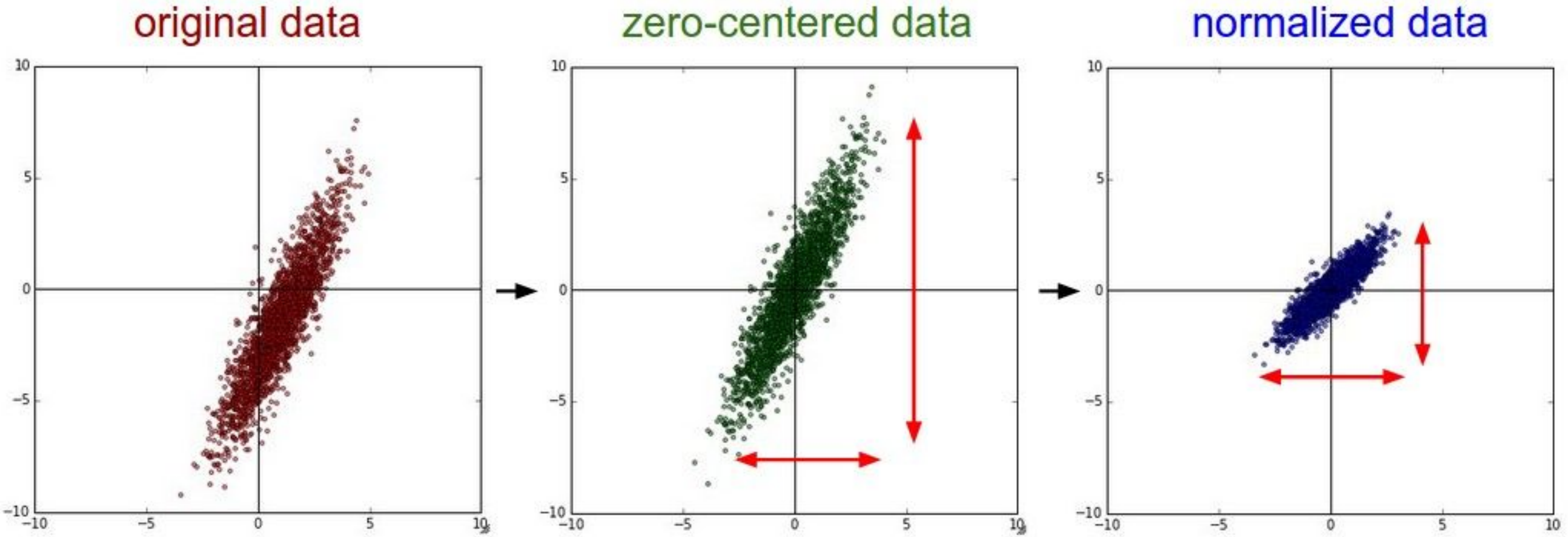
Keras tutorial

Inputs

Grayscale image: (200,100)

RGB image: (200,100,3)

Batch of RGB images: (256,200,100,3)


original data
zero-centered data
normalized data
10
5
0
−5
−10
−10
−5
0
5
10

```python
batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)
```

```python
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```python
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
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(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

```python
epochs=1000
for e in range(epochs):
    model.fit(xtrain, ytrain,
              shuffle=False, epochs=1,
              batch_size=256,
              validation_data=(xval,yval))

#   checkpointing
    if e%10==0:
        model.save('traintop-'+str(e)+'-'+str(e)+'.h5')
```

```
xtrain = np.load(direc+'bottleneck-train-'+str(experimentNumber)+'-'+str(validationFold)+'-0.npy')
ytrain = np.load(direc+'ytrain-'+str(experimentNumber)+'-'+str(validationFold)+'-0.npy')

i = Input(shape=xtrain.shape[1:])
a = Flatten(name='a1')(i)
a= Dense(256, activation='relu',name='a2')(a)
a = Dropout(0.6,name='a3')(a)
o = Dense(ytrain.shape[1], activation='softmax',name='a4')(a)
model = Model(inputs=i,outputs=o)

opt=keras.optimizers.RMSprop(lr=0.00001, rho=0.9, epsilon=None, decay=0.0)
model.compile(optimizer=opt,
          loss='categorical_crossentropy',
          metrics=['categorical_accuracy'])

print (model.summary())
```

```python
epochs=1000
for e in range(epochs):
    for batch_counter in range(trains):
        batch_num=batch_counter
        fname=direc+'bottleneck-train-'+str(experimentNumber)+'-'+str(validationFold)+'-'+str(batch_num)+'.npy'
        fname2=direc+'ytrain-'+str(experimentNumber)+'-'+str(validationFold)+'-'+str(batch_num)+'.npy'

        xtrain = np.load(fname)
        ytrain = np.load(fname2)
        print (str(batch_counter)+'/'+str(trains-1))
        print (xtrain.shape,ytrain.shape)
        model.fit(xtrain, ytrain,batch_size=128)

    if e%10==0:
        model.save(direc+'traintop-'+str(experimentNumber)+'-'+str(validationFold)+'-'+str(e)+'.h5')
```

Deploying to phone

AT&T 1:40 PM 76%

3
99% it's Hemangioma

Select image

MyModel

AT&T 1:40 PM 76%

4
99% it's Nevus

Select image

MyModel

AT&T 1:40 PM 75%

5
99% it's Hemangioma

Select image

MyModel

AT&T 1:39 PM 76%

1
80% it's Capillary Malformation

Select image

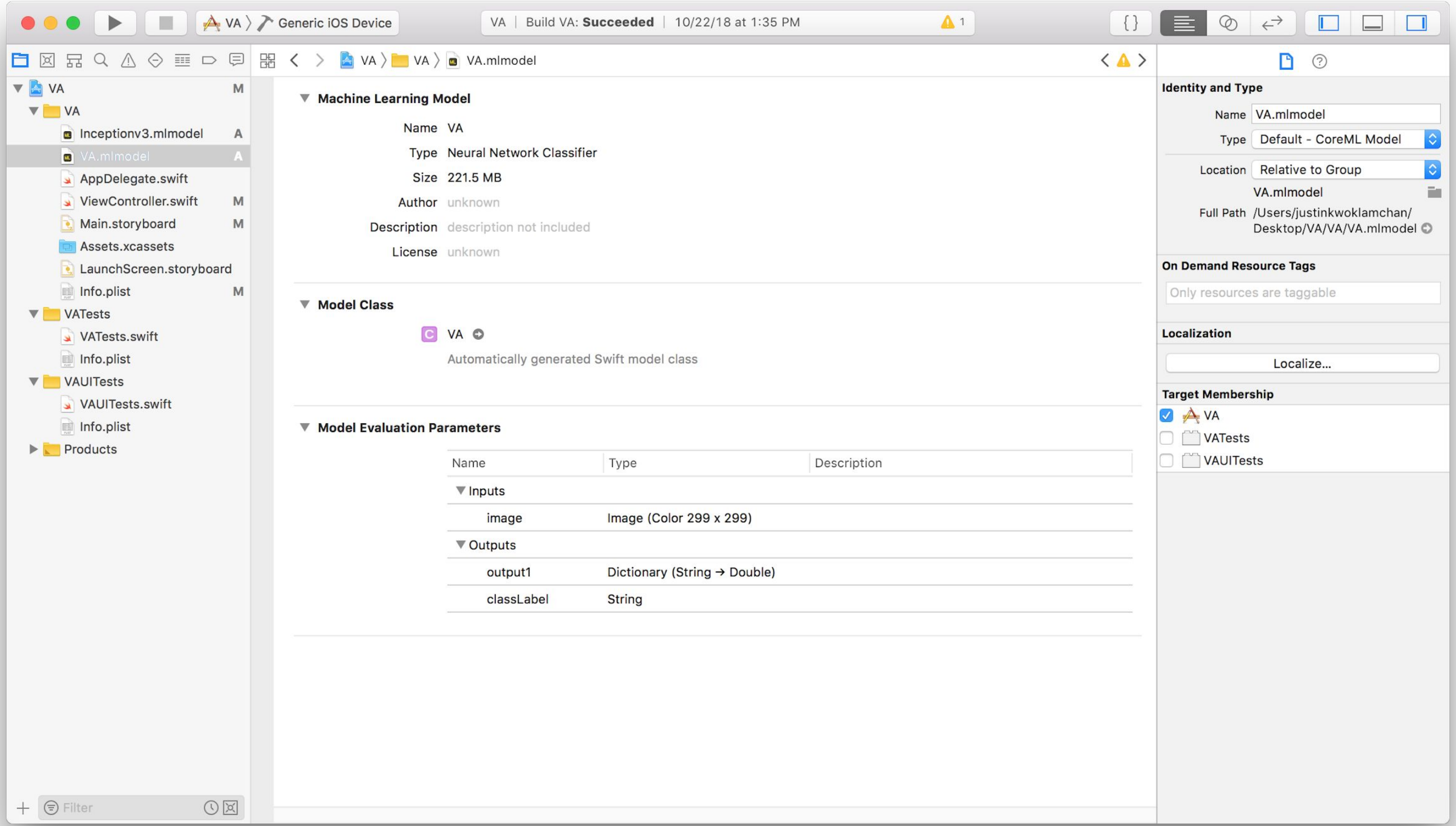
MyModel

Load in the .h5 model file

```
from keras.applications.inception_v3 import InceptionV3
from keras.layers import Dense, Flatten, Dropout
from keras.models import Model

base_model = InceptionV3(include_top=False, weights='imagenet', input_shape=(299,299,3))

i = base_model.output
a = Flatten(name='a1')(i)
a = Dense(256,activation='relu',name='a2')(a)
a = Dropout(.6,name='a3')(a)
o = Dense(9, activation='softmax',name='a4')(a)
model = Model(inputs=base_model.input, outputs=o)

model.load_weights('traintop-4-1-80.h5',by_name=True)
```

Convert to .mlmodel file

```
class_labels=['Hemangioma','Pyogenic Granuloma','Venous Malformation','Capillary Malformation',
              'Spider Angioma','Lymphatic Malformation','Atopic Dermatitis','Milia','Nevus']

import coremltools
coreml_model = coremltools.converters.keras.convert(
    model,
    input_names="image",
    image_input_names="image",
    image_scale=1/127.5,
    red_bias=-1.0,
    green_bias=-1.0,
    blue_bias=-1.0,
    class_labels=class_labels,
    )
coreml_model.save('VA.mlmodel')
```


VA
Generic iOS Device
VA | Build VA: Succeeded | 10/22/18 at 1:35 PM
VA
VA
Inceptionv3.mlmodel
VA.mlmodel
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
VATests
VATests.swift
Info.plist
VAUITests
VAUITests.swift
Info.plist
Products
Filter
VA.mlmodel
Machine Learning Model
Name VA
Type Neural Network Classifier
Size 221.5 MB
Author unknown
Description description not included
License unknown
Model Class
VA
Automatically generated Swift model class
Model Evaluation Parameters
Name
Type
Description
Inputs
image
Image (Color 299 x 299)
Outputs
output1
Dictionary (String → Double)
classLabel
String
Identity and Type
Name VA.mlmodel
Type Default - CoreML Model
Location Relative to Group
VA.mlmodel
Full Path /Users/justinkwoklamchan/Desktop/VA/VA/VA.mlmodel
On Demand Resource Tags
Only resources are taggable
Localization
Localize...
Target Membership
VA
VATests
VAUITests

```swift
func classify(image: CIImage) {
    let model:VNCoreMLModel = getmodel()

    let request = VNCoreMLRequest(model: model) { [weak self] request, error in
        guard let results = request.results as? [VNClassificationObservation],
            let topResult = results.first else {
                fatalError("unexpected result type from VNCoreMLRequest")
        }

        for i in 0..<9 {
            print("\(results[i].identifier) \(results[i].confidence)")
        }

        DispatchQueue.main.async { [weak self] in
            let t = (self?.counter)!
            self?.answerLabel.text = "\(t)\n\(Int(topResult.confidence * 100))% it's \(topResult.identifier)"
        }
    }

    request.imageCropAndScaleOption = .centerCrop

    let handler = VNImageRequestHandler(ciImage: image)
    DispatchQueue.global(qos: .userInteractive).async {
        do {
            try handler.perform([request])
        } catch {
            print(error)
        }
    }
}
```

```swift
func readimg() -> CIImage? {
    let filePath = Bundle.main.resourcePath!+"/tree.png"

    do {
        let image = UIImage(contentsOfFile: filePath);
        guard let ciImage = CIImage(image: image!) else {
            fatalError("couldn't convert UIImage to CIImage")
        }
        return ciImage
    }
}
```