Neural Networks (NN)

From CS231, 2017, Stanford
Neural networks: Architectures

“Fully-connected” layers

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

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

**Sigmoid**
\[\sigma(x) = \frac{1}{1+e^{-x}}\]

**tanh**
\[\tanh(x)\]

**ReLU**
\[\text{max}(0, x)\]

**Leaky ReLU**
\[\text{max}(0.1x, x)\]

**Maxout**
\[\text{max}(w_1^T x + b_1, w_2^T x + b_2)\]

**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

Deep Learning Framework Power Scores 2018

- TensorFlow: 96.77
- Keras: 51.55
- PyTorch: 22.72
- Caffe: 17.15
- Theano: 12.02
- MXNET: 8.37
- CNTK: 4.89
- DeepLearning4j: 3.65
- Caffe2: 2.71
- Chainer: 1.18
- FastAI: 1.06
Convolutional Neural Networks (CNN)
A bit of history:
Gradient-based learning applied to document recognition
[LeCun, Bottou, Bengio, Haffner 1998]
A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]
Fast-forward to today: ConvNets are everywhere

Classification

Retrieval

Fast-forward to today: ConvNets are everywhere.

**Detection**

![Detection](image1)

**Segmentation**

![Segmentation](image2)

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

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


[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

Minor errors

A man in a baseball uniform throwing a ball

Somewhat related

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

A woman standing on a beach holding a surfboard

All images are CC0 Public domain:

Captions generated by Justin Johnson using NeuralTalk2
Fast-forward to today: ConvNets are everywhere

[Simonyan et al. 2014]

Figures copyright Simonyan et al., 2014. Reproduced with permission.

[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

input

$Wx$

10 x 3072 weights

activation

1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)
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**

32x32x3 image

5x5x3 filter

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

Filters always extend the full depth of the input volume
Convolution Layer

A 32x32x3 image
5x5x3 filter $w$

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

32x32x3 image
5x5x3 filter
convolve (slide) over all spatial locations

activation map

32
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

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation maps

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.

- **CONV, ReLU**
  - e.g. 6 filters of size 5x5x3
  - e.g. 10 filters of size 5x5x6
Keras Convolutional Neural Network

```python
define_model:
    model = Sequential()
    model.add(Conv2D(6, (5, 5), activation='relu', input_shape=(32, 32, 3)))
    model.add(Conv2D(10, (5, 5), activation='relu'))
```

Pooling layer
- makes the representations smaller and more manageable
- operates over each activation map independently:
MAX POOLING

Single depth slice

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

max pool with 2x2 filters and stride 2

<table>
<thead>
<tr>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
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

This image by Paolo Guereta is licensed under CC-BY 2.0
Regularization: Dropout

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

Optimizers

MNIST Multilayer Neural Network + dropout

- AdaGrad
- RMSProp
- SGDNesterov
- AdaDelta
- Adam

training cost vs iterations over entire dataset
Keras Full Convolutional Neural Network

```python
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)
2. Neural Architecture Search (NASNet)

Diagram showing the architecture search process with nodes for Number of Filters, Filter Height, Filter Width, Stride Height, Stride Width, Number of Filters, and Filter Height.
Comparison

![Comparison graph showing the accuracy (precision @1) against the number of Mult-Add operations (millions) for various models like NASNet, Inception, ResNet, PolyNet, DPN, SENet, ShuffleNet, MobileNet, and InceptionNet. The graph illustrates the trade-off between model accuracy and computational efficiency.]
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)
```