W4995 Applied Machine Learning

Advanced Neural Networks

04/22/19

Andreas C. Müller

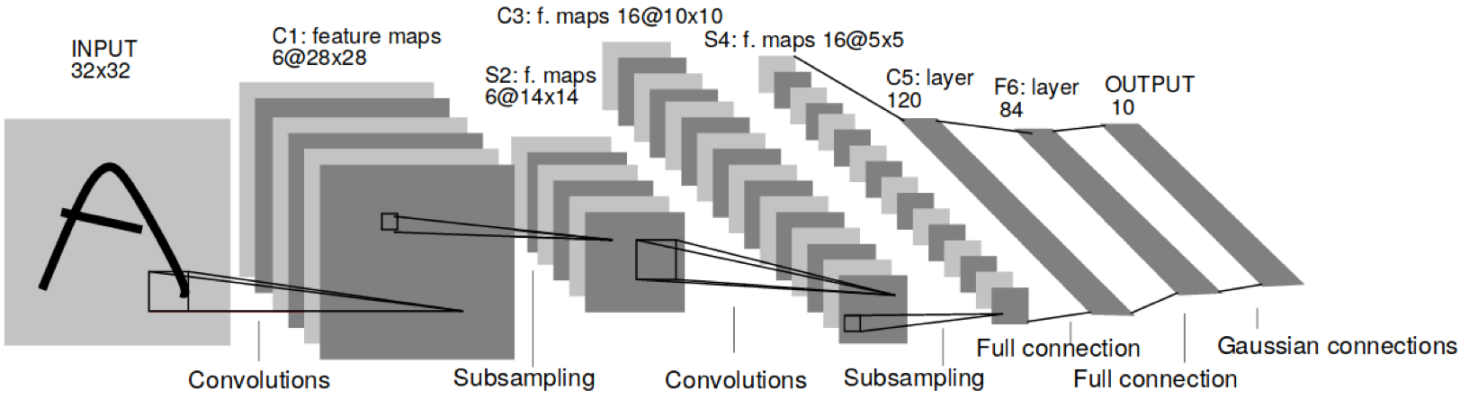
Definition of Convolution

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

$$= \sum_{m=-\infty}^{\infty} f[n-m]g[m]$$

f 0 1 2 0 1 1 0 0 1 0

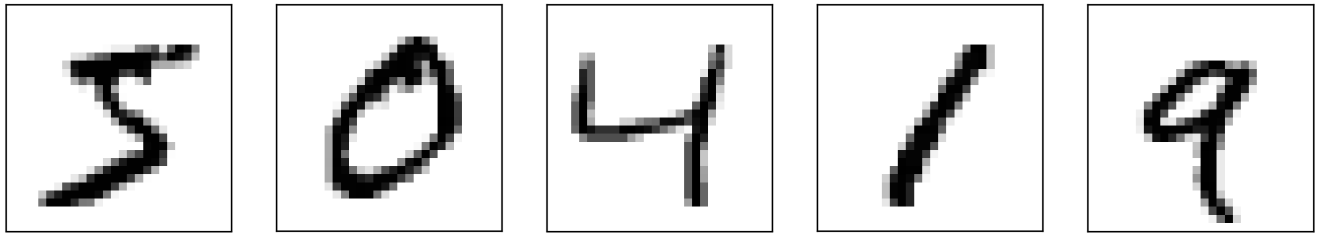
Convolution Neural Networks



- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner: Gradient-based learning applied to document recognition

Convnets vs Fully Connected Nets Illustrated

MNIST and Permuted MNIST



```
rng = np.random.RandomState(42)
perm = rng.permutation(784)
X_train_perm = X_train.reshape(-1, 784)[:, perm].reshape(-1, 28, 28)
X_test_perm = X_test.reshape(-1, 784)[:, perm].reshape(-1, 28, 28)
```

Fully Connected vs Convolutional

```
model = Sequential([
    Dense(512, input_shape=(784,), activation='relu'),
    Dense(10, activation='softmax'),
])
model.compile("adam", "categorical_crossentropy",
              metrics=['accuracy'])
```

```
_________________________________________________________
Layer (type)          Output Shape         Param #
=========================================================
dense_7 (Dense)       (None, 512)           401920
_________________________________________________________
dense_8 (Dense)       (None, 10)            5130
=========================================================
Total params: 407,050
Trainable params: 407,050
_________________________________________________________
```

Fully Connected vs Convolutional

```
model = Sequential([
    Dense(512, input_shape=(784,), activation='relu'),
    Dense(10, activation='softmax'),
])
model.compile("adam", "categorical_crossentropy",
              metrics=['accuracy'])
```

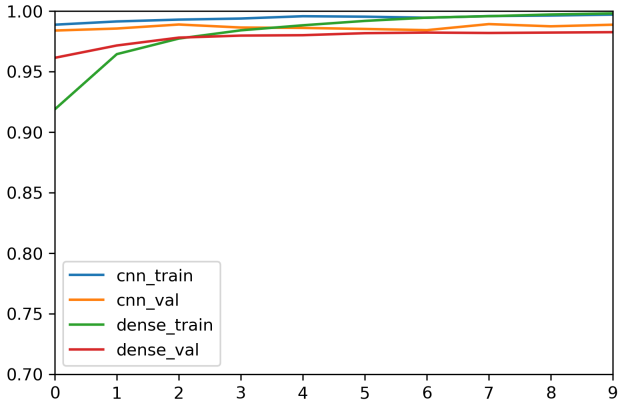
```
_________________________________________________________
Layer (type)          Output Shape         Param #
=========================================================
dense_7 (Dense)       (None, 512)           401920
_________________________________________________________
dense_8 (Dense)       (None, 10)            5130
=========================================================
Total params: 407,050
Trainable params: 407,050
_________________________________________________________
```

```
num_classes = 10
cnn = Sequential()
cnn.add(Conv2D(32, kernel_size=(3, 3),
               activation='relu',
               input_shape=input_shape))
cnn.add(MaxPooling2D(pool_size=(2, 2)))
cnn.add(Conv2D(32, (3, 3), activation='relu'))
cnn.add(MaxPooling2D(pool_size=(2, 2)))
cnn.add(Flatten())
cnn.add(Dense(64, activation='relu'))
cnn.add(Dense(num_classes, activation='softmax'))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 800)               0
_________________________________________________________________
dense_9 (Dense)              (None, 64)                51264
_________________________________________________________________
dense_10 (Dense)             (None, 10)                650
=================================================================
Total params: 61,482
Trainable params: 61,482
```

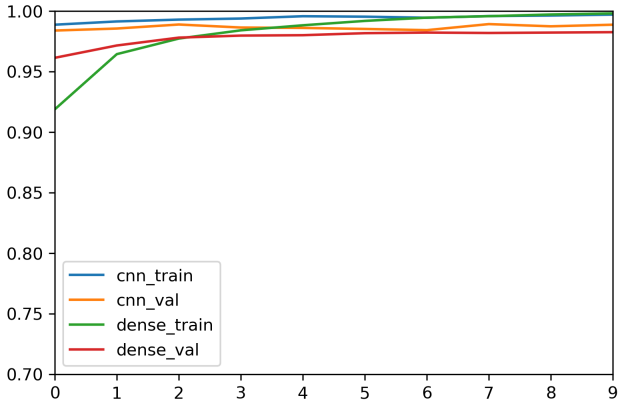
Training curves

Original Data

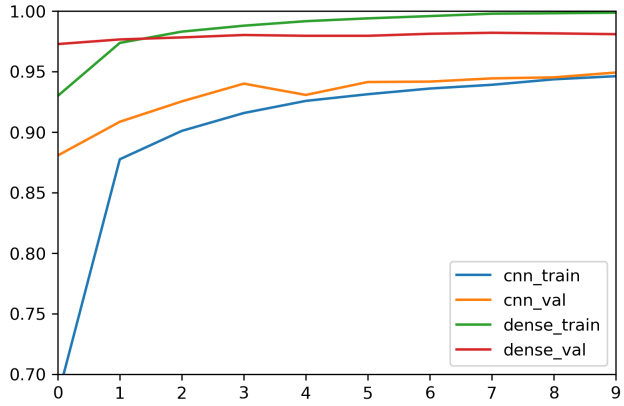


Training curves

Original Data



Shuffled Data



Residual Neural Networks

Problem

Solution

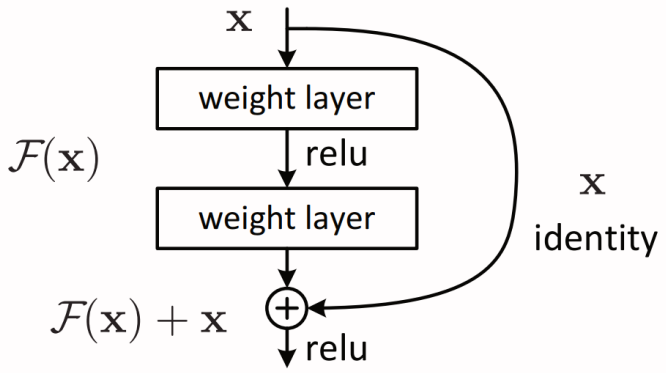


Figure 2. Residual learning: a building block.

$$y = F(x, \{W_i\}) + x \quad \text{for same size layers}$$

Solution

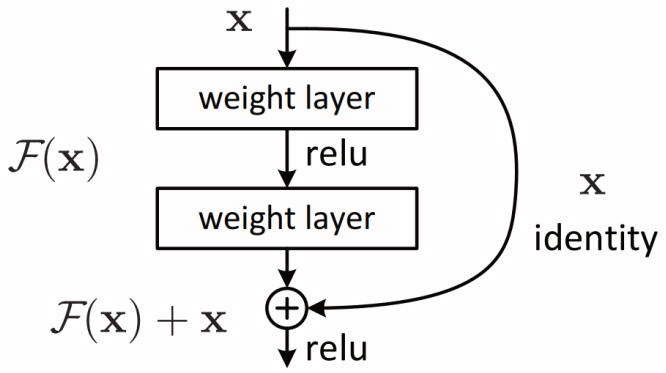
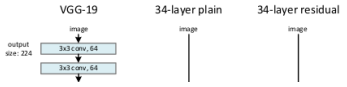


Figure 2. Residual learning: a building block.

$$y = F(x, \{W_i\}) + x \quad \text{for same size layers}$$

$$y = F(x, \{W_i\}) + W_s x \quad \text{for different size layers}$$


VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64

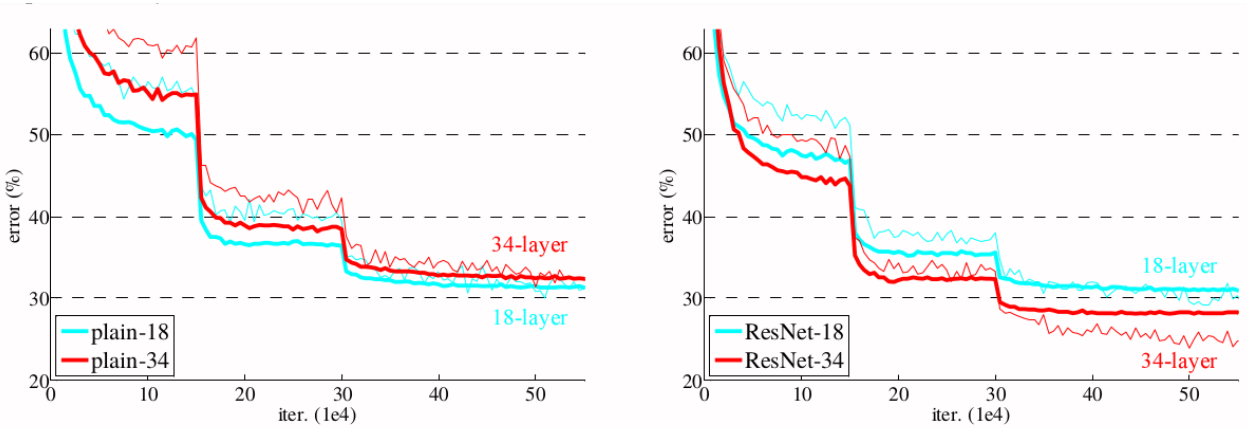


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

This was Dec 2015. Current SOTA: 15.7% top-1 error

https://paperswithcode.com/sota/image-classification-on-imagenet

ResNets in Keras

Requires functinal API: https://keras.io/getting-started/functional-api-guide/

```
from keras.layers import Input, Dense
from keras.models import Model

# This returns a tensor
inputs = Input(shape=(784,))

# a layer instance is callable on a tensor, and returns a tensor
x = Dense(64, activation='relu')(inputs)
x = Dense(64, activation='relu')(x)
predictions = Dense(10, activation='softmax')(x)

# This creates a model that includes
# the Input layer and three Dense layers
model = Model(inputs=inputs, outputs=predictions)
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(data, labels)  # starts training
```

CNN with Functional API

```
from keras.layers import Input, Conv2D, MaxPooling2D, Flatten
from keras.models import Model

num_classes = 10
inputs = Input(shape=(28, 28, 1))
conv1_1 = Conv2D(32, (3, 3), activation='relu',
                 padding='same')(inputs)
conv1_2 = Conv2D(32, (3, 3), activation='relu',
                 padding='same')(conv1_1)
maxpool1 = MaxPooling2D(pool_size=(2, 2))(conv1_2)
conv2_1 = Conv2D(32, (3, 3), activation='relu',
                 padding='same')(maxpool1)
conv2_2 = Conv2D(32, (3, 3), activation='relu',
                 padding='same')(conv2_1)
maxpool2 = MaxPooling2D(pool_size=(2, 2))(conv2_2)
flat = Flatten()(maxpool2)
dense = Dense(64, activation='relu')(flat)
predictions = Dense(num_classes, activation='softmax')(dense)

model = Model(inputs=inputs, outputs=predictions)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         (None, 28, 28, 1)         0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 28, 28, 32)        9248
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 14, 32)        9248
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 14, 14, 32)        9248
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1568)              0
_________________________________________________________________
dense_13 (Dense)             (None, 64)                100416
_________________________________________________________________
dense_14 (Dense)             (None, 10)                650
=================================================================
Total params: 129,130
Trainable params: 129,130
Non-trainable params: 0
_________________________________________________________________
```

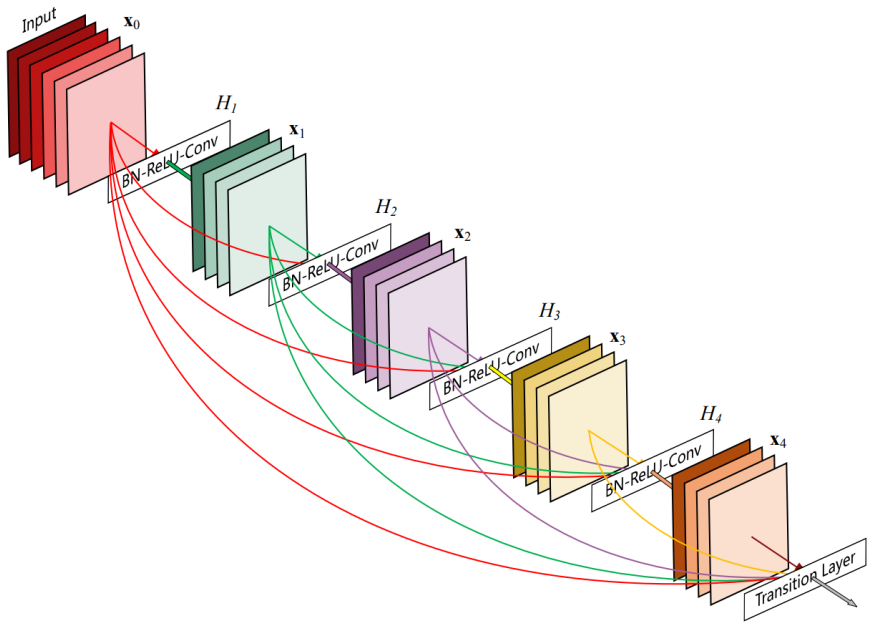
Adding Skip connections

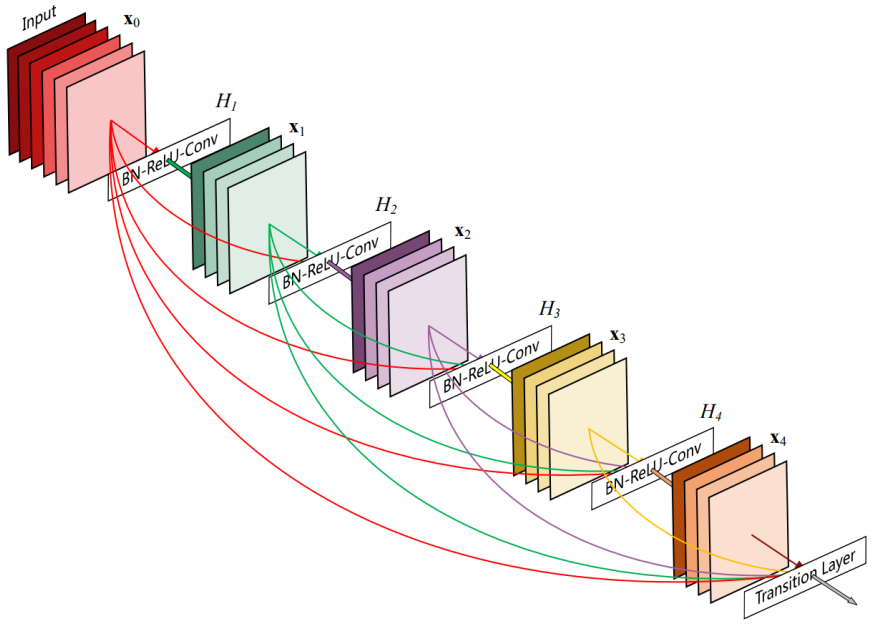
```
from keras.layers import Input, Conv2D, MaxPooling2D, Flatten
from keras.models import Model

num_classes = 10
inputs = Input(shape=(28, 28, 1))
conv1_1 = Conv2D(32, (3, 3), activation='relu',
                 padding='same')(inputs)
conv1_2 = Conv2D(32, (3, 3), activation='relu',
                 padding='same')(conv1_1)
maxpool1 = MaxPooling2D(pool_size=(2, 2))(conv1_2)
conv2_1 = Conv2D(32, (3, 3), activation='relu',
                 padding='same')(maxpool1)
conv2_2 = Conv2D(32, (3, 3), activation='relu',
                 padding='same')(conv2_1)
# Actually doesn't work, this net is too small
skip2 = add([maxpool1, conv2_2])
maxpool2 = MaxPooling2D(pool_size=(2, 2))(skip2)
flat = Flatten()(maxpool2)
dense = Dense(64, activation='relu')(flat)
predictions = Dense(num_classes, activation='softmax')(dense)

model = Model(inputs=inputs, outputs=predictions)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #     Connecte
=================================================================
input_14 (InputLayer)        (None, 28, 28, 1)         0
_________________________________________________________________
conv2d_57 (Conv2D)           (None, 28, 28, 32)        320         input_14
_________________________________________________________________
conv2d_58 (Conv2D)           (None, 28, 28, 32)        9248        conv2d_5
_________________________________________________________________
max_pooling2d_23 (MaxPooling2D) (None, 14, 14, 32)     0           conv2d_5
_________________________________________________________________
conv2d_59 (Conv2D)           (None, 14, 14, 32)        9248        max_pool
_________________________________________________________________
conv2d_60 (Conv2D)           (None, 14, 14, 32)        9248        conv2d_5
_________________________________________________________________
add_11 (Add)                 (None, 14, 14, 32)        0           max_pool
                                                                   conv2d_6
_________________________________________________________________
max_pooling2d_24 (MaxPooling2D) (None, 7, 7, 32)       0           add_11[0
_________________________________________________________________
flatten_14 (Flatten)         (None, 1568)              0           max_pool
_________________________________________________________________
dense_35 (Dense)             (None, 64)                100416      flatten_
_________________________________________________________________
dense_36 (Dense)             (None, 10)                650         dense_35
=================================================================
Total params: 129,130
Trainable params: 129,130
Non-trainable params: 0
_________________________________________________________________
```

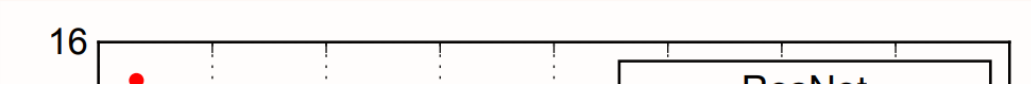
Densely Connected Convolutional Networks (DenseNet)

Huang et. al - Densely Connected Convolutional Networks (2016)


Input
$\mathbf{x}_0$
H_1
BN-ReLU-Conv
$\mathbf{x}_1$
H_2
BN-ReLU-Conv
$\mathbf{x}_2$
H_3
BN-ReLU-Conv
$\mathbf{x}_3$
H_4
BN-ReLU-Conv
$\mathbf{x}_4$
Transition Layer


Input
$\mathbf{x}_0$
H_1
BN-ReLU-Conv
$\mathbf{x}_1$
H_2
BN-ReLU-Conv
$\mathbf{x}_2$
H_3
BN-ReLU-Conv
$\mathbf{x}_3$
H_4
BN-ReLU-Conv
$\mathbf{x}_4$
Transition Layer


Input
Dense Block 1
Dense Block 2
Dense Block 3
Prediction

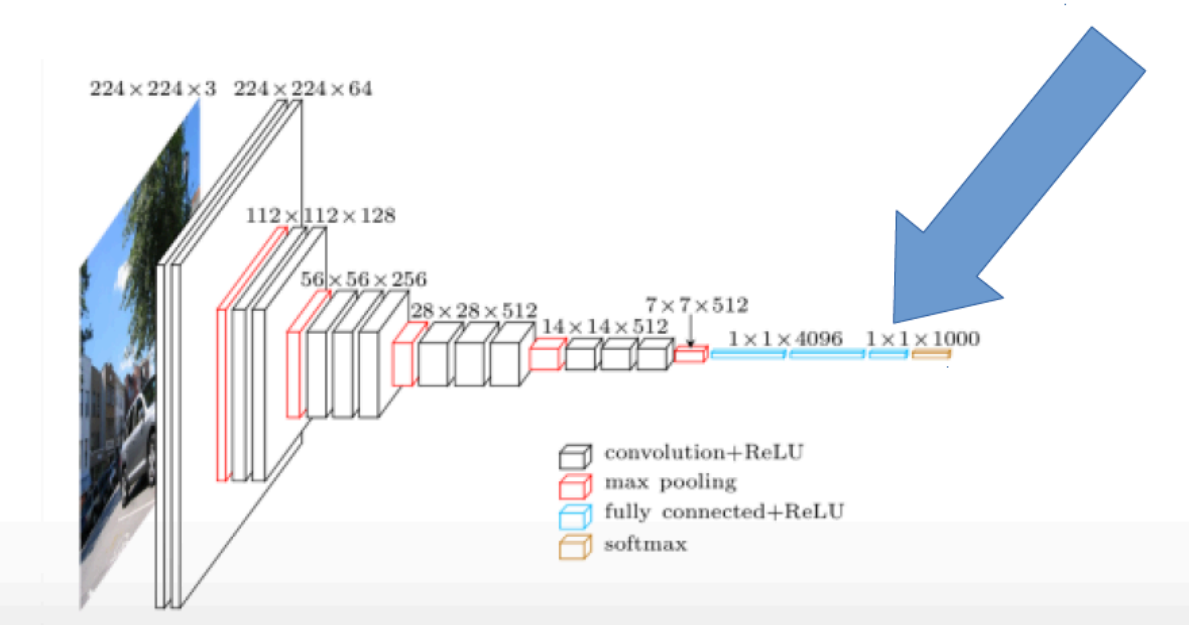

16

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88
DenseNet121	33 MB	0.745	0.918	8,062,504	121
DenseNet169	57 MB	0.759	0.928	14,307,880	169
DenseNet201	80 MB	0.770	0.933	20,242,984	201

https://keras.io/applications/

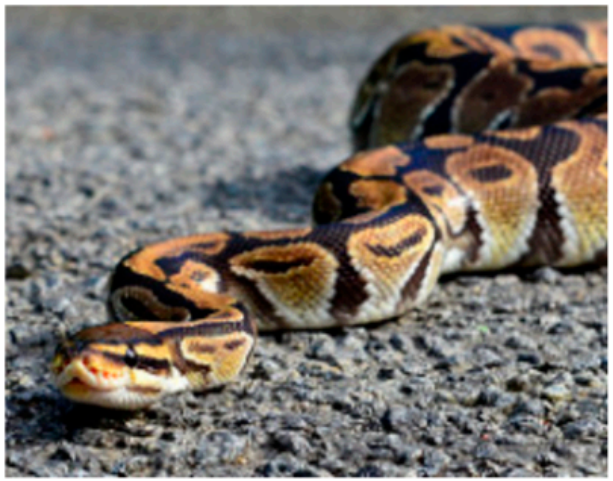
Transfer learning

Transfer Learning



See http://cs231n.github.io/transfer-learning/

Ball snake vs Carpet Python



```
import flickrapi
import json
flickr = flickrapi.FlickrAPI(api_key, api_secret, format='json')
json.loads(flickr.photos.licenses.getInfo().decode("utf-8"))
def get_url(photo_id="33510015330"):
    response = flickr.photos.getsizes(photo_id=photo_id)
    sizes = json.loads(response.decode('utf-8'))['sizes']['size']
    for size in sizes:
        if size['label'] == "Small":
            return size['source']

get_url()
ids = search_ids("ball snake", per_page=100)
urls_ball = [get_url(photo_id=i) for i in ids]

from urllib.request import urlretrieve
import os
for url in urls_carpet:
    urlretrieve(url, os.path.join("snakes", "carpet", os.path.basename(url)))
```


Carpet Python (100 total)
Ball Snake (100 total)

Extracting Features using VGG

```
from keras.preprocessing import image


X = np.array([image.img_to_array(img) for img in images_carpet + images_ball])

# load VGG16
model = applications.VGG16(include_top=False, weights='imagenet')

# preprocessing for VGG16
from keras.applications.vgg16 import preprocess_input

X_pre = preprocess_input(X)
features = model.predict(X_pre)
print(X.shape)
print(features.shape)
features_ = features.reshape(200, -1)
```

```
(200, 224, 224, 3)
```

```
(200, 7, 7, 512)
```

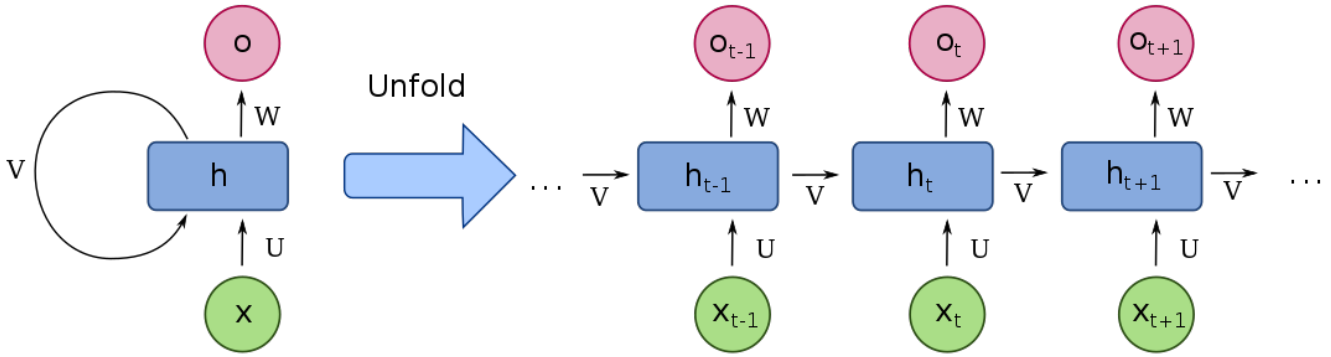
Classification with LogReg

```
from sklearn.linear_model import LogisticRegressionCV
lr = LogisticRegressionCV().fit(X_train, y_train)
print(lr.score(X_train, y_train))
print(lr.score(X_test, y_test))
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, lr.predict(X_test))
```

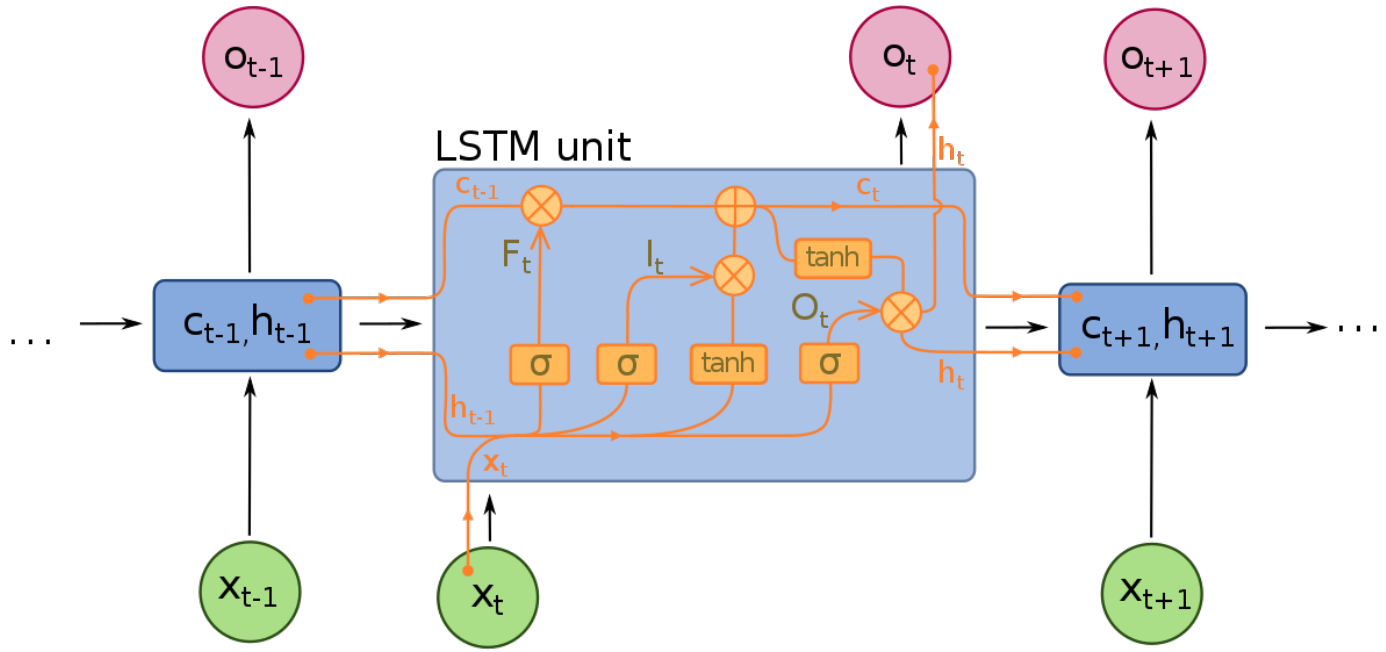
```
1.0
0.82
array([[24,  1],
       [ 8, 17]])
```

Finetuning

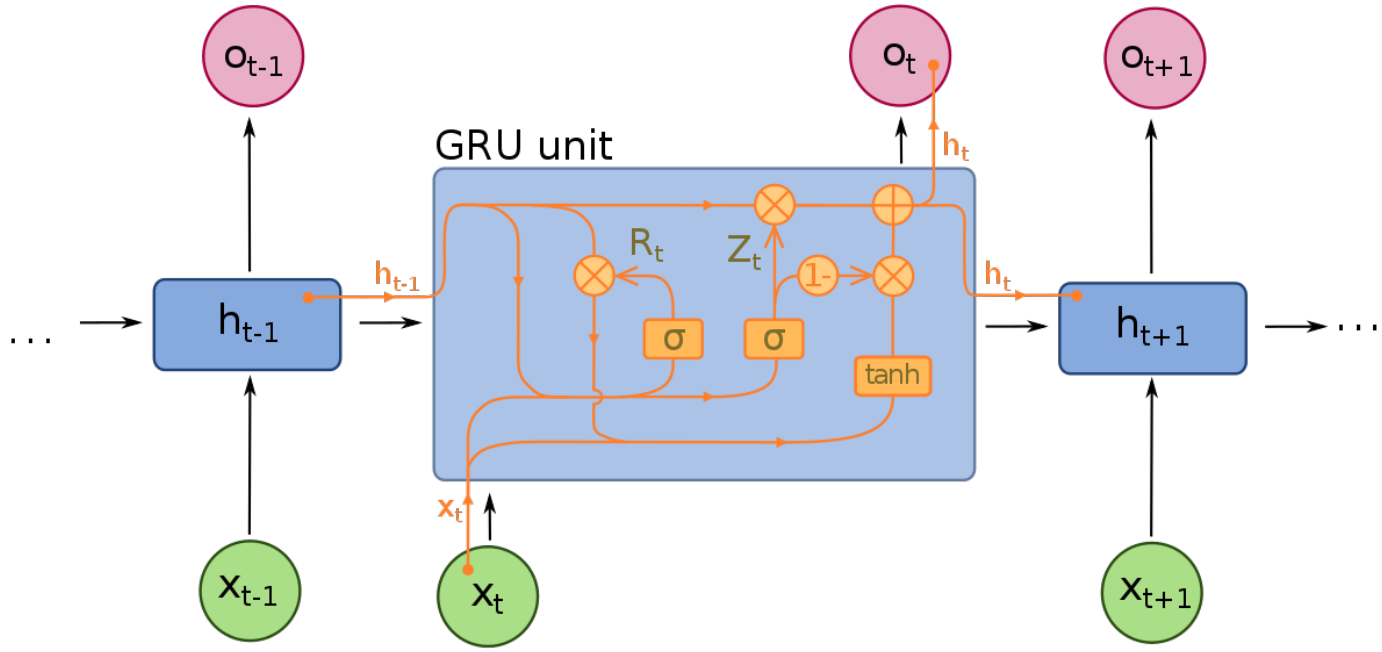
Recurrent Neural Networks



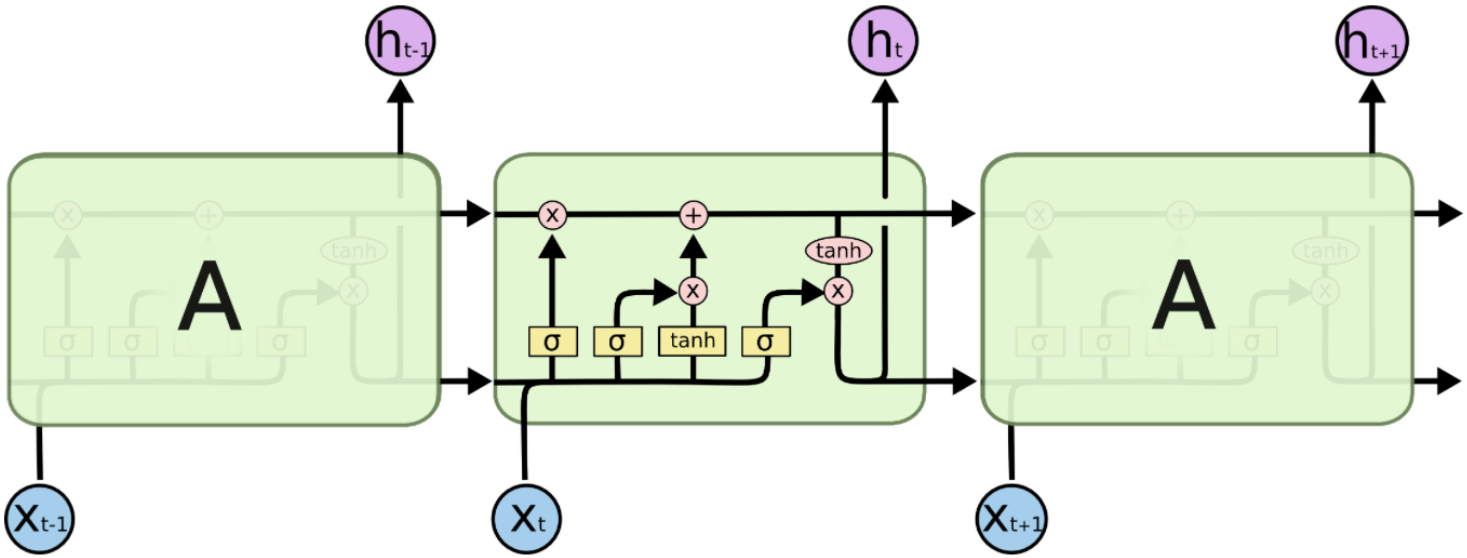
Images from http://colah.github.io/posts/2015-08-Understanding-LSTMs/



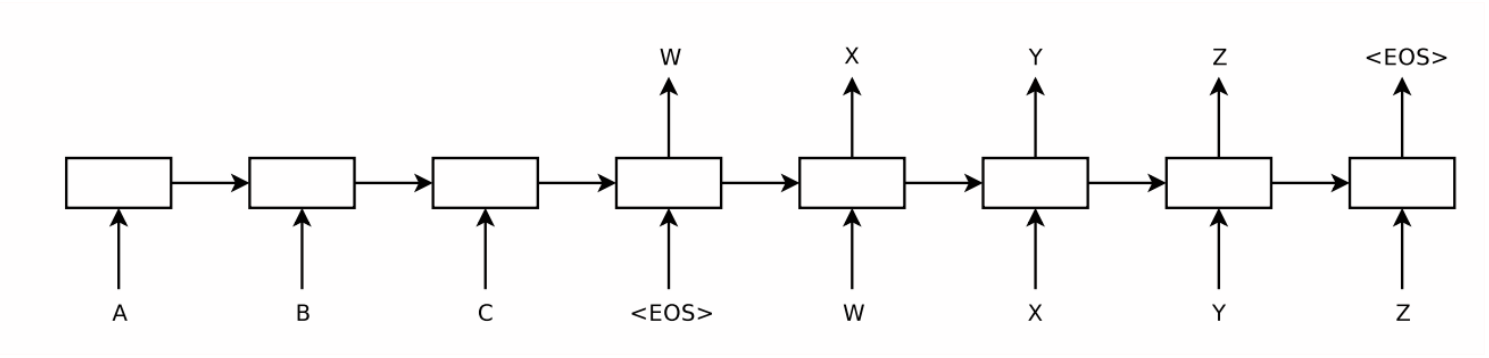
Hochreiter, Schmidhuber - Long Short-Term Memory (1997)



Cho et. al. - Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation(2014)

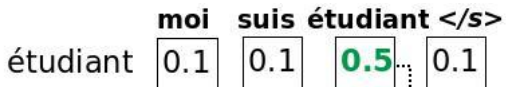


The repeating module in an LSTM contains four interacting layers.



Sutskever et. al. - Sequence to Sequence Learning with Neural Networks (2014)

Machine Translation



Question answering

Dataset	Example	Article / Paragraph
SQuAD	**Q**: How many provinces did the Ottoman empire contain in the 17th century? **A**: 32	**Article**: Ottoman Empire **Paragraph**: ... At the beginning of the 17th century the empire contained 32 provinces and numerous vassal states. Some of these were later absorbed into the Ottoman Empire, while others were granted various types of autonomy during the course of centuries.
CuratedTREC	**Q**: What U.S. state's motto is "Live free or Die"? **A**: New Hampshire	**Article**: Live Free or Die **Paragraph**: "Live Free or Die" is the official motto of the U.S. state of New Hampshire, adopted by the state in 1945. It is possibly the best-known of all state mottos, partly because it conveys an assertive independence historically found in American political philosophy and partly because of its contrast to the milder sentiments found in other state mottos.
WebQuestions	**Q**: What part of the atom did Chadwick discover?† **A**: neutron	**Article**: Atom **Paragraph**: ... The atomic mass of these isotopes varied by integer amounts, called the whole number rule. The explanation for these different isotopes awaited the discovery of the neutron, an uncharged particle with a mass similar to the proton, by the physicist James Chadwick in 1932. ...
WikiMovies	**Q**: Who wrote the film Gigli? **A**: Martin Brest	**Article**: Gigli **Paragraph**: Gigli is a 2003 American romantic comedy film written and directed by Martin Brest and starring Ben Affleck, Jennifer Lopez, Justin Bartha, Al Pacino, Christopher Walken, and Lainie Kazan.

Chen et. al. - Reading Wikipedia to Answer Open-Domain Questions https://rajpurkar.github.io/SQuAD-explorer/

41 / 43

Adverserial Samples

(a) (b)

Right column: correctly classified image, left column classified as Ostrich. Center: difference.

Szegedy et. al.: Intriguing properties of neural networks

Questions ?