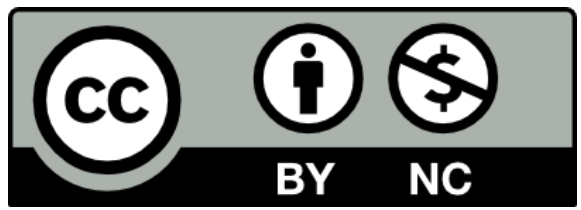
Business 4720 - Class 16

Convolutional Neural Networks using Python

Joerg Evermann

Faculty of Business Administration
Memorial University of Newfoundland
jevermann@mun.ca



MEMORIAL UNIVERSITY

This Class

What You Will Learn:

- Deep Learning Concepts
  - Convolutional Neural Networks
  - Text Classification with Neural Networks

MEMORIAL UNIVERSITY

Based On

Gareth James, Daniel Witten, Trevor Hastie and Robert Tibshirani: *An Introduction to Statistical Learning with Applications in R*. 2nd edition, corrected printing, June 2023. (ISLR2)

`https://www.statlearning.com`

Chapter 10

Kevin P. Murphy: *Probabilistic Machine Learning – An Introduction*. MIT Press 2022.

`https://probml.github.io/pml-book/book1.html`

Chapter 13, 14, 15

Tensorflow Guides

`https://www.tensorflow.org/guide`
`https://www.tensorflow.org/tutorials/load_data/text`

MEMORIAL UNIVERSITY

Convolutional Neural Networks

Task

- Image classification

Problem

- Translation, scaling, and rotation of objects in images

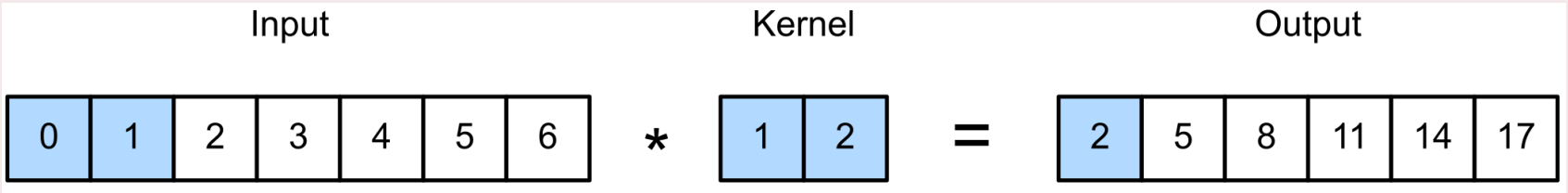
Solution

- Learning low-level and high-level ”features” by learning a sequence of ”filters” or ”kernels”

MEMORIAL UNIVERSITY

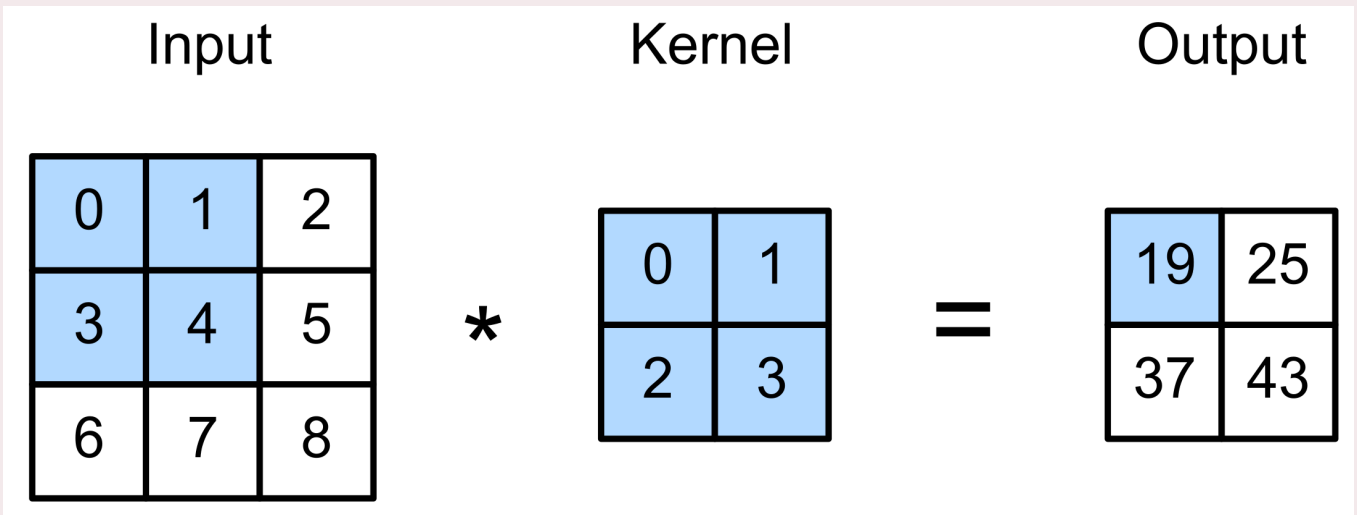
Convolution

1-Dimensional



Source: Murphy Fig. 14.4

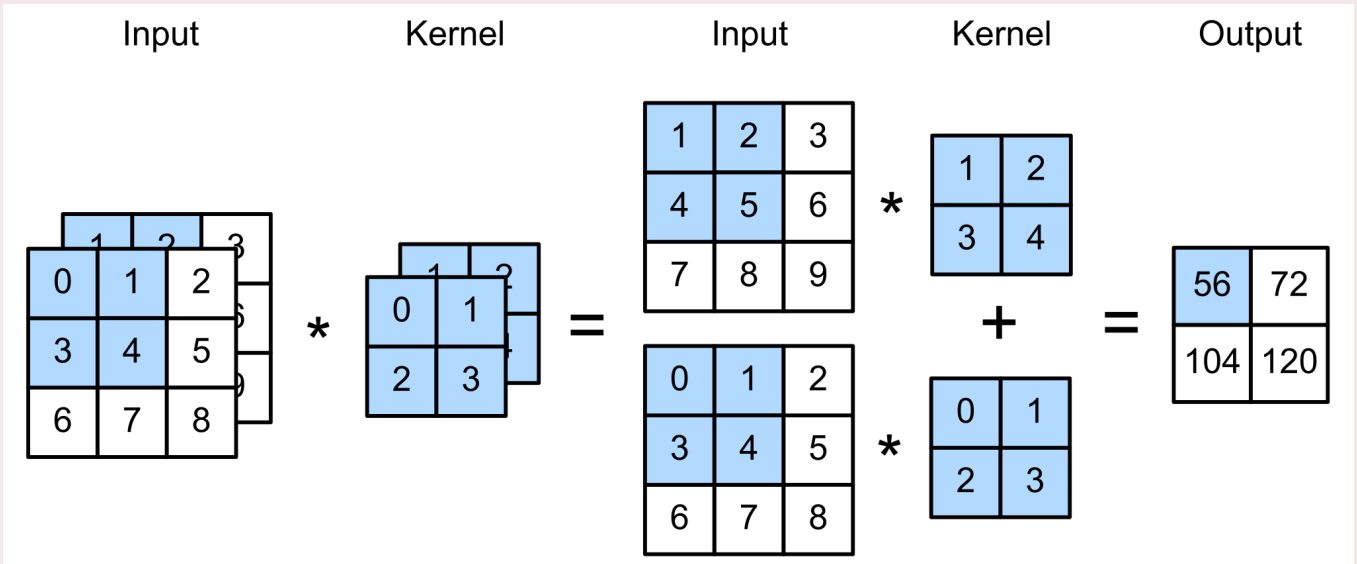
2-Dimensional



Source: Murphy Fig. 14.5

Convolution [cont'd]

Multi-channel 2D Convolution



Source: Murphy Fig. 14.9

In a convolutional neural net layer, the convolution kernels are ***trainable parameters***.

CNN Parameters

- Let the kernel be of size $k \times l$
- Let the input be of m channels
- Let the output be n channels
- Then, there are $k \times l$ elements for each of the m input dimensions, i.e. a total of $m \times k \times l$
- Then, there are $m \times k \times l$ elements for each of the n output dimensions, i.e. a total of $n \times m \times k \times l$
- Additionally, there are n bias terms, one for each output channel

Example: Let the kernel size be 5×5 applied to an input with 3 channels and an output of 32 channels. Then, the CNN layer has $32 \times 3 \times 5 \times 5 + 32 = 2432$ parameters.

MEMORIAL UNIVERSITY

Hands-On Exercises

Calculate the number of trainable parameters for the following CNN layers:

1. A kernel of size 4×4 operating on an input with 3 channels and an output of 8 channels.
2. A kernel of size 2×2 operating on an input with 8 channels and an output of 16 channels.
3. A kernel of size 3×3 operating on an input with 16 channels and an output of 4 channels.
4. A kernel of size 5×5 operating on an input with 4 channels and an output with 4 channels.

Assume the previous four layers are part of the same neural network, what is the total number of trainable parameters?

MEMORIAL UNIVERSITY

Striding

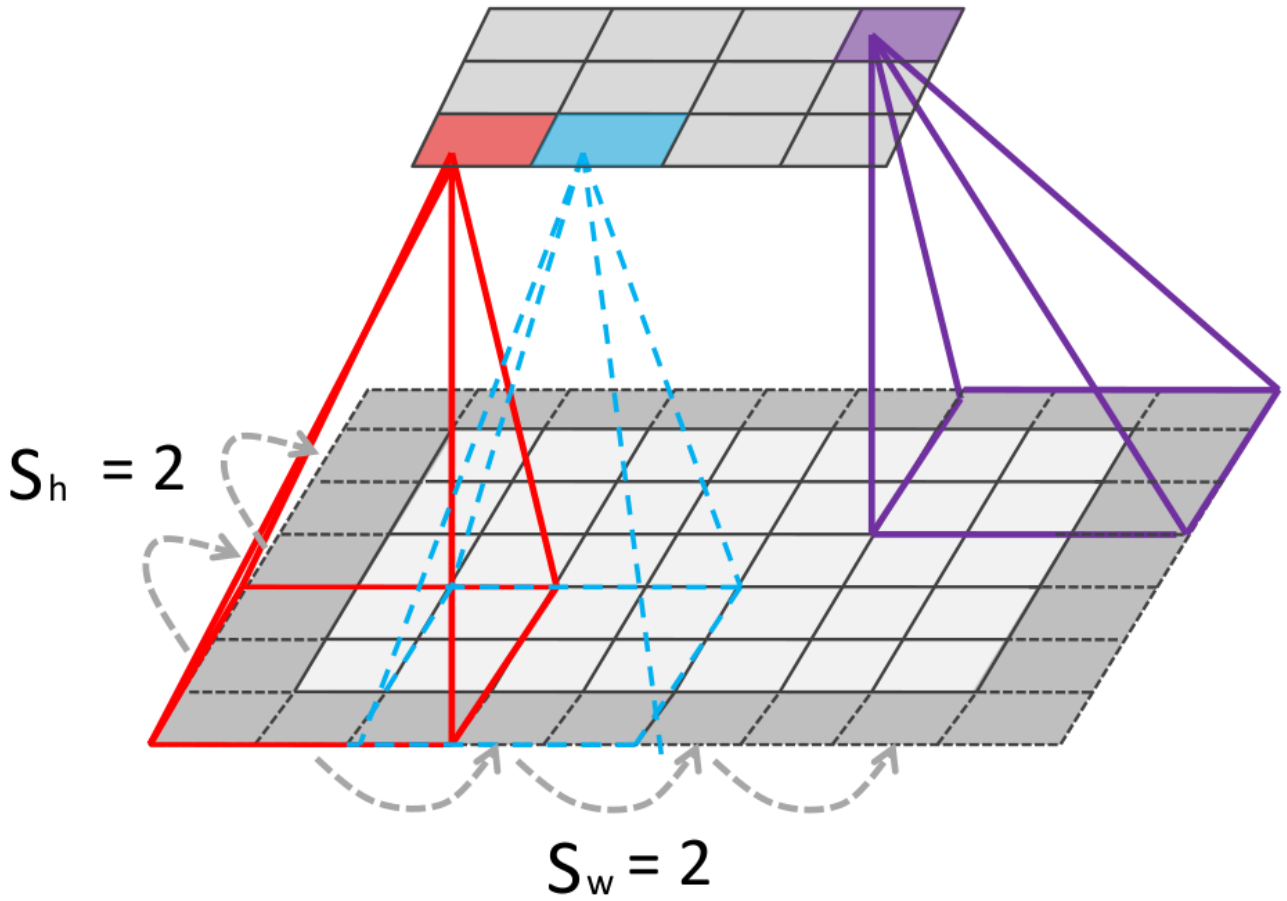
- Take "larger steps" moving the kernel
- Remove redundancy
- Increase efficiency

Padding

- Surround the data matrix by zeros
- Increases weight of edge pixels
- Allows same-convolution, i.e. output size is the same as input size

Convolution [cont'd]

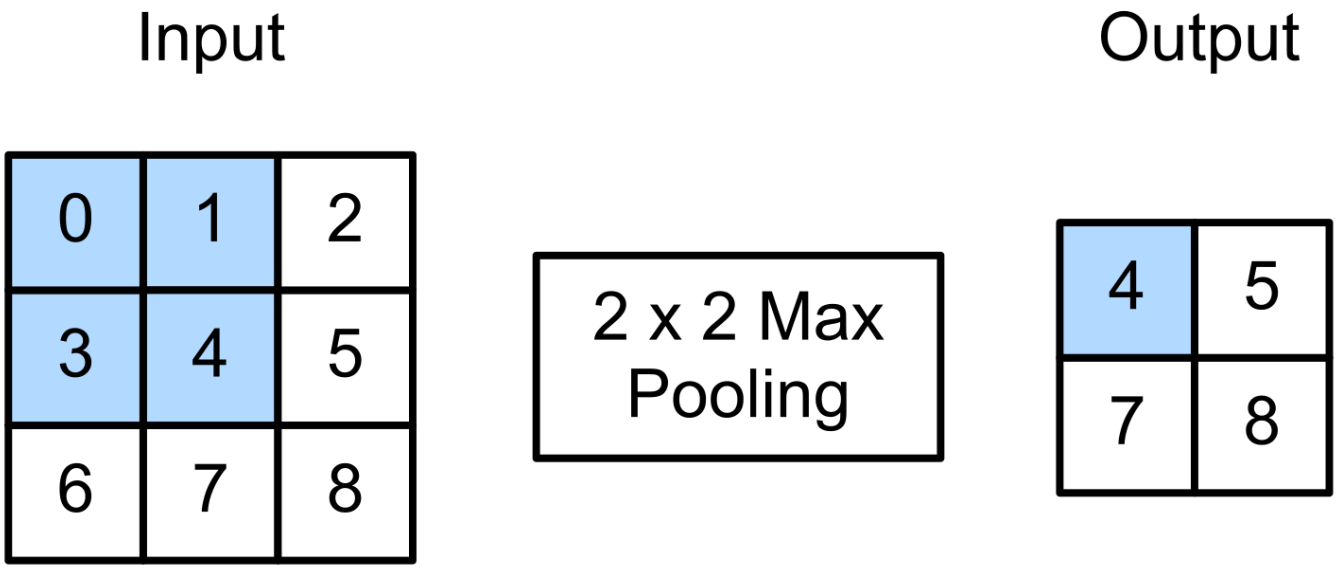
Striding and Padding



Source: Murphy Fig. 14.8(b)

Pooling

- Recognize location independent features
- Max pooling or average pooling
- Pooling layers have no trainable parameters



Source: Murphy Fig. 14.12

Hands-On Exercises

1. Assume the following 5×5 input matrix:

$$\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 3 & 2 & 1 & 3 \\ 3 & 2 & 1 & 3 & 2 \\ 2 & 1 & 3 & 2 & 1 \\ 1 & 3 & 2 & 1 & 3 \end{bmatrix}$$

and the following 3×3 convolution kernel:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

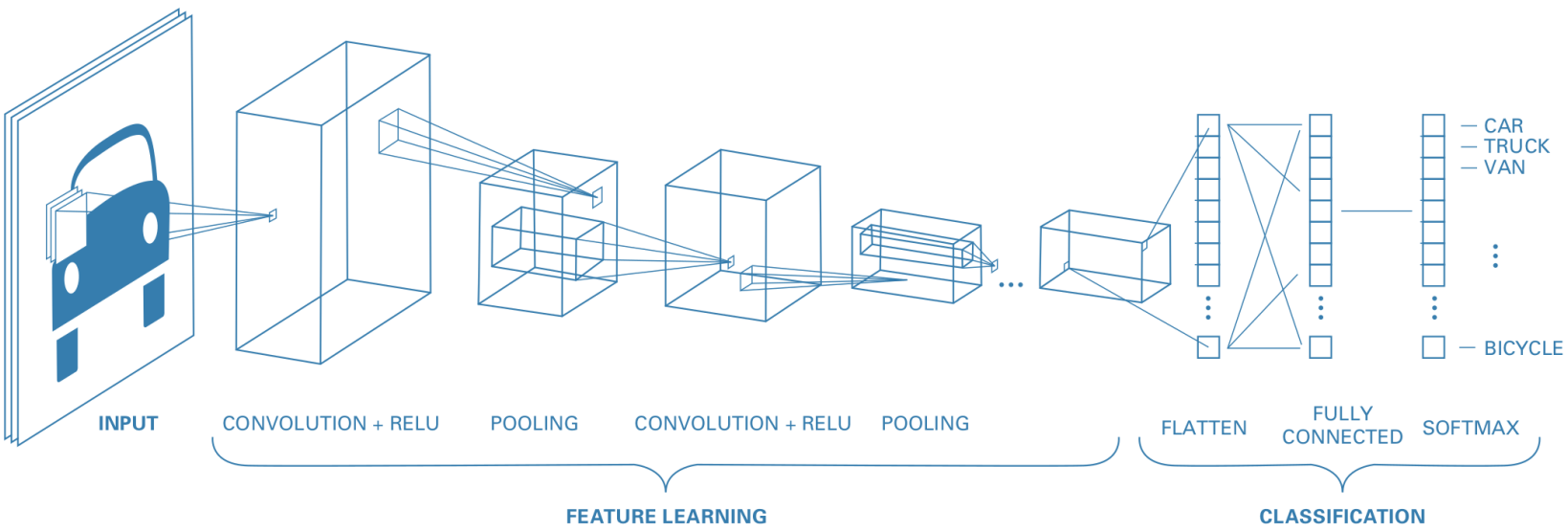
Zero-pad the matrix with two zeros on all sides and using stride 2, calculate the convolution result. What are the dimensions of the convolution output?

2. Apply a 2×2 max pooling layer to the result of the previous exercise.

MEMORIAL UNIVERSITY

CNN

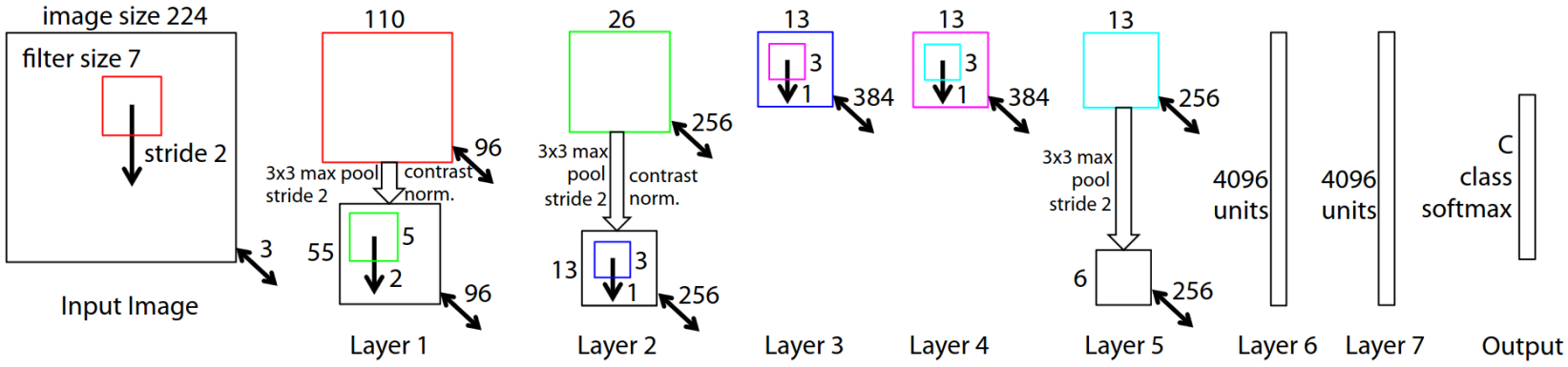
- Multiple convolution and pooling (subsampling) layers
- Flatten final feature maps
- Multiple fully connected layers
- Softmax output



Source: Murphy Fig. 14.13

CNN

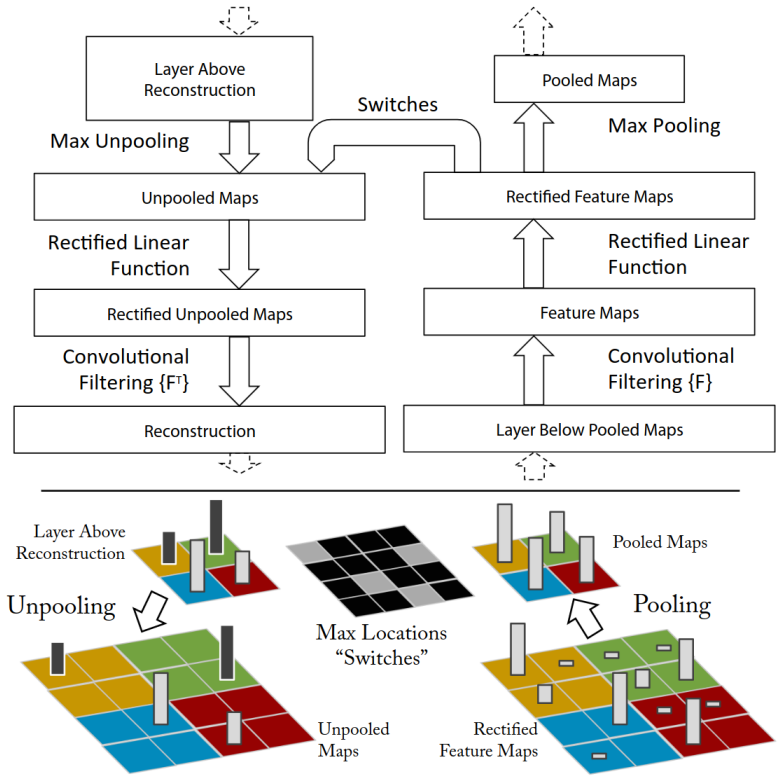
- Multiple convolutional kernels applied in parallel to the same image (”channels”)



Source: Matthew D. Zeiler and Rob Fergus (2013) Visualizing and Understanding Convolutional Networks. https://doi.org/10.48550/arXiv.1311.2901

CNN – Feature Maps

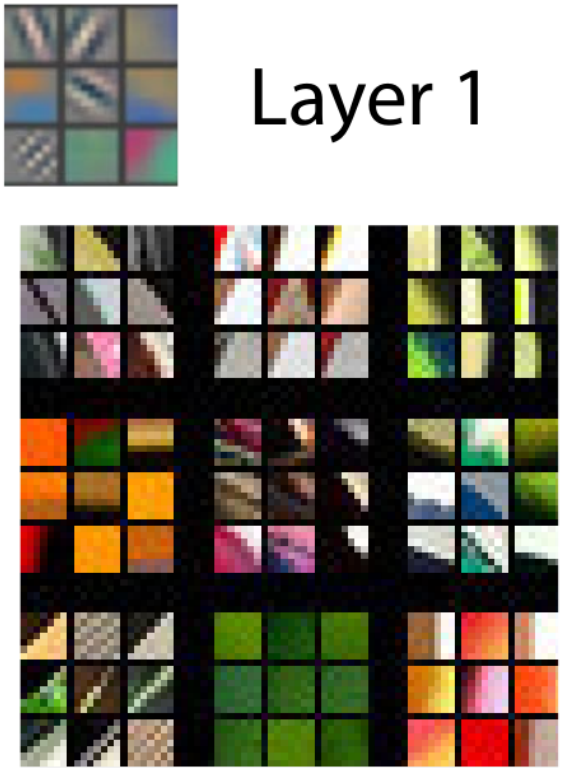
- ReLU'd and Pooled Convolution is a "**Feature Map**"
- Features maps at different layers identify different types of features
- Features form the input to the fully connected layers for classification
- Visualization e.g. with DeconvNet



Source: Matthew D. Zeiler and Rob Fergus (2013) Visualizing and Understanding Convolutional Networks. https://doi.org/10.48550/arXiv.1311.2901

MEMORIAL UNIVERSITY

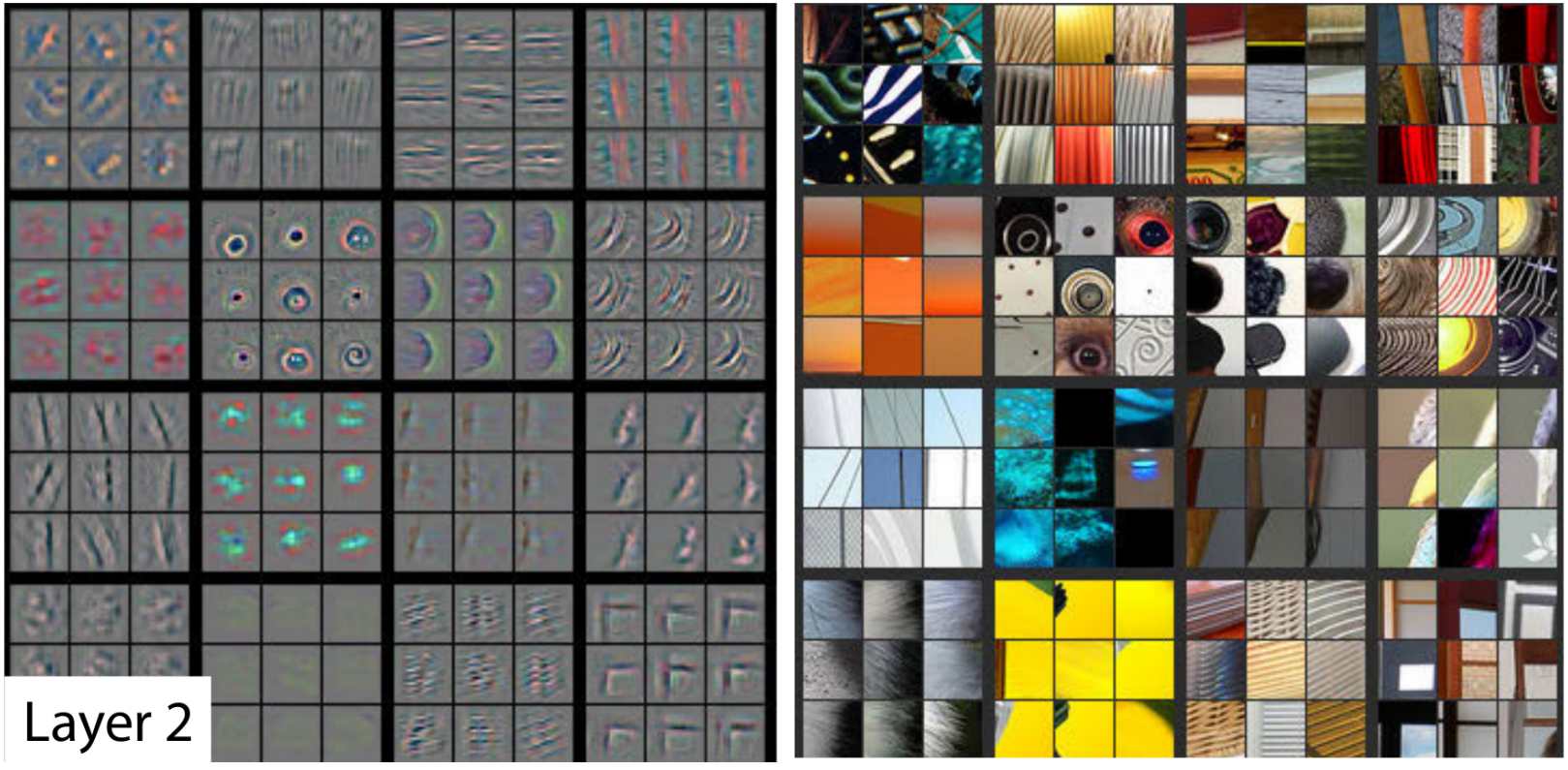
CNN – Feature Maps



Source: Matthew D. Zeiler and Rob Fergus (2013) Visualizing and Understanding Convolutional Networks. https://doi.org/10.48550/arXiv.1311.2901

MEMORIAL UNIVERSITY

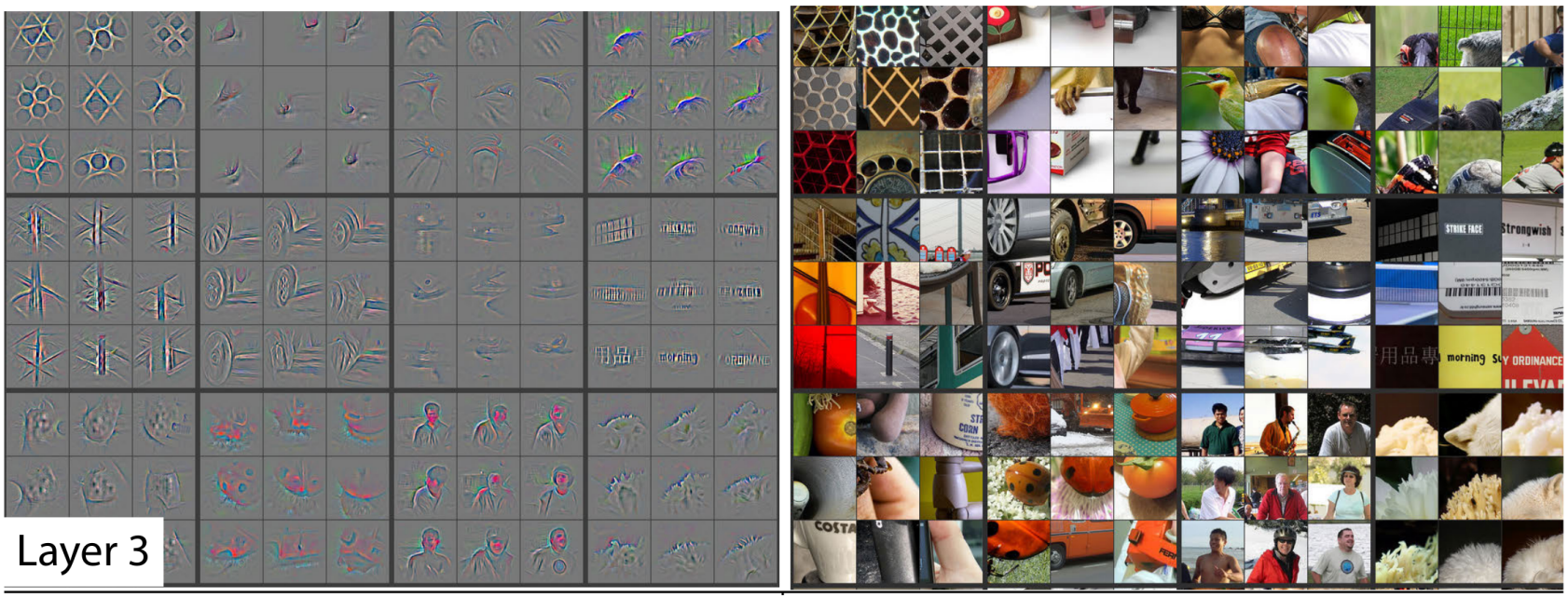
CNN – Feature Maps



Source: Matthew D. Zeiler and Rob Fergus (2013) Visualizing and Understanding Convolutional Networks. https://doi.org/10.48550/arXiv.1311.2901

MEMORIAL UNIVERSITY

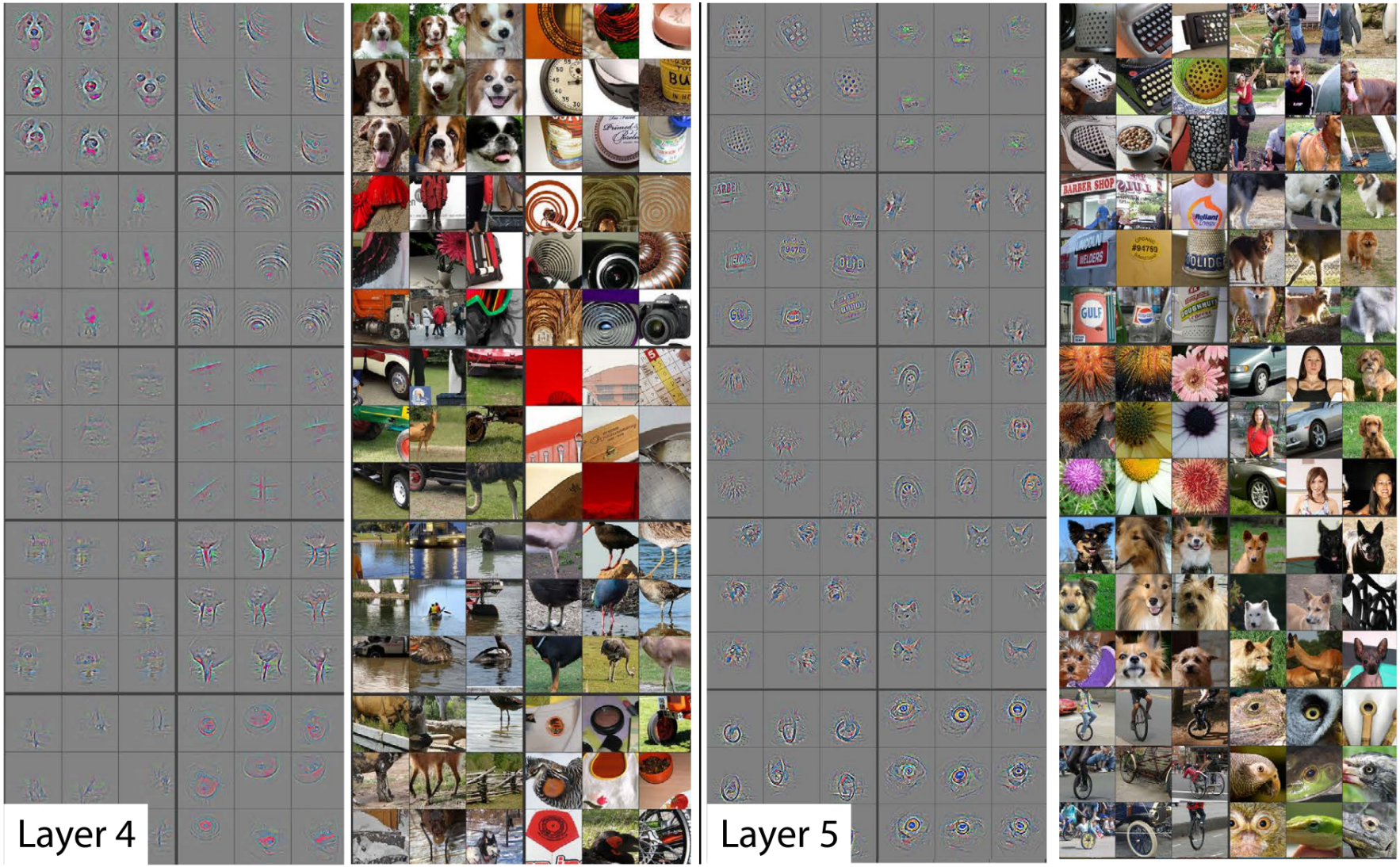
CNN – Feature Maps



Source: Matthew D. Zeiler and Rob Fergus (2013) Visualizing and Understanding Convolutional Networks. https://doi.org/10.48550/arXiv.1311.2901

MEMORIAL UNIVERSITY

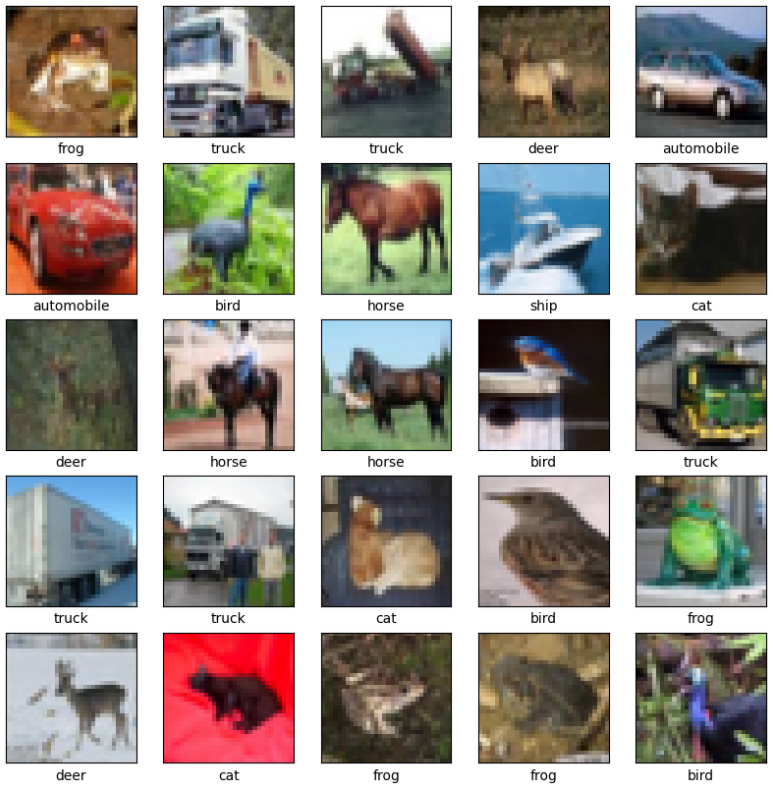
CNN – Feature Maps



Source: Matthew D. Zeiler and Rob Fergus (2013) Visualizing and Understanding Convolutional Networks. https://doi.org/10.48550/arXiv.1311.2901

MEMORIAL UNIVERSITY

CNN in Keras



- 60,000 images, 32×32 resolution, 3 channels
- 10 classes
- Latest error rate $< 0.5\%$

MEMORIAL UNIVERSITY

CNN in Keras

Import the data:

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import datetime

(train_images, train_labels), (test_images, test_labels) \
    = datasets.cifar10.load_data()

# Normalize pixel values to be between 0 and 1
train_images = train_images / 255.0
test_images = test_images / 255.0
```

Show sample images:

```
import plotly
import plotly.express as px

fig = plotly.subplots.make_subplots(rows=5, cols=5)
for i in range(25):
    fig.add_trace(px.imshow(train_images[i]).data[0],
                  row=i//5+1, col=i%5+1)
fig.show()
```

CNN in Keras

Create a simple convolutional model:

```
model = models.Sequential()

model.add(layers.Conv2D(8, (3, 3), activation='relu', \
    input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(16, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(16, (3, 3), activation='relu'))
```

Add dense (fully-connected) layers for classification. There are 10 target classes:

```
model.add(layers.Flatten())
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(10))

model.summary()
```

CNN in Keras

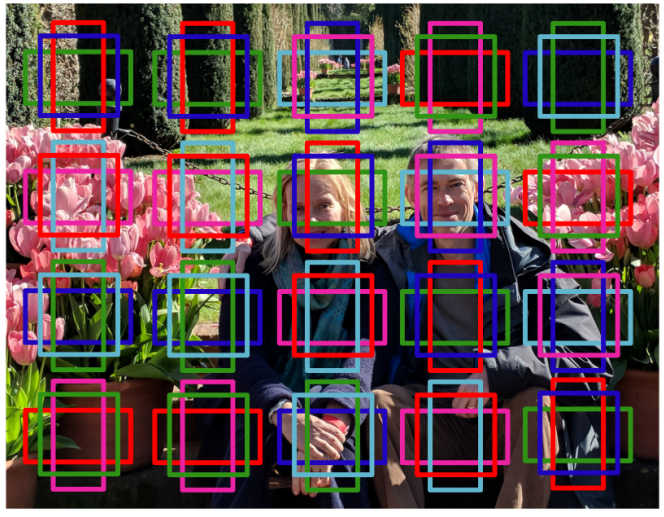
Compile and train the model:

```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses \
        .SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'])

model.fit(
    train_images, train_labels, epochs=10,
    validation_data=(test_images, test_labels))
```

MEMORIAL UNIVERSITY

Other Computer Vision Tasks for CNNs

Object Detection



Source: Murphy Fig. 14.27

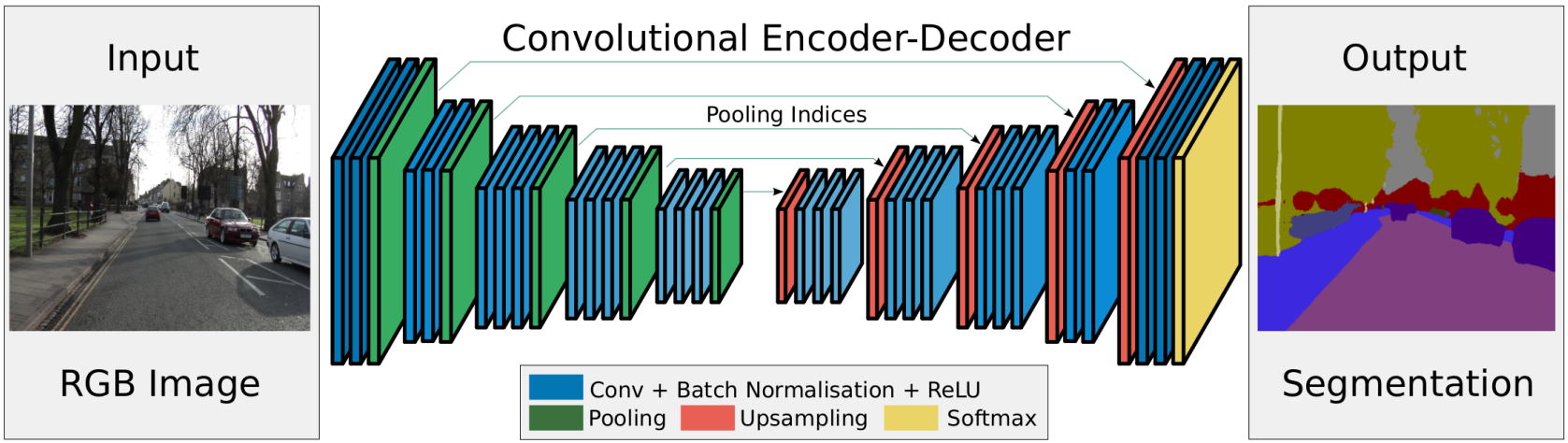
$$f_\theta : R^{H \times W \times K} \rightarrow [0,1]^{A \times A} \times \{1, \ldots, C\}^{A \times A} \times (R^4)^{A \times A}$$

For each anchor box, learn the object presence probability, the object category and two offset vectors to be added to the center of the box to the top left and bottom right corners.

MEMORIAL UNIVERSITY

Other Computer Vision Tasks for CNNs

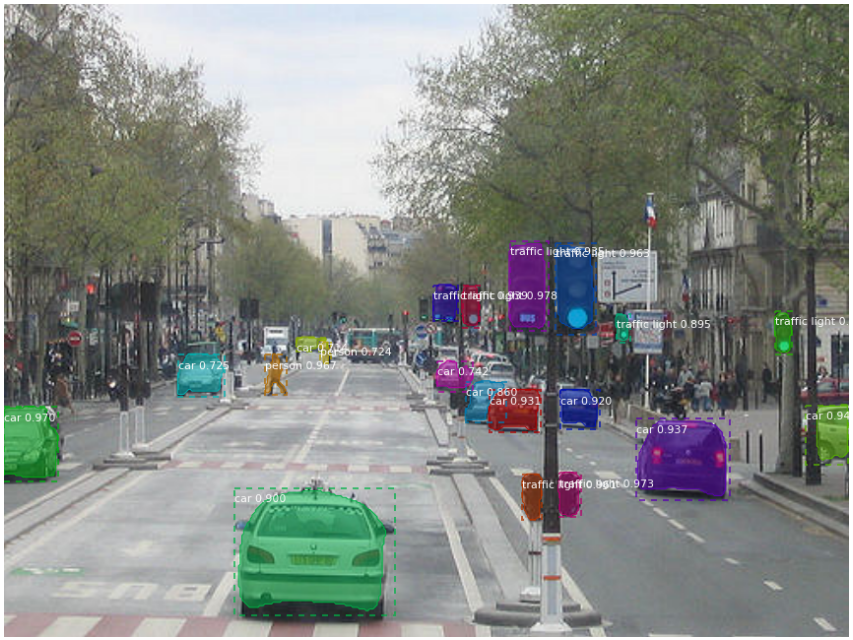
Semantic Segmentation



Source: Murphy Fig. 14.29

Using an **encoder-decoder** or **U-net** network to identify the class of each pixel in an image.

Other Computer Vision Tasks for CNNs



`https://github.com/matterport/Mask_RCNN`

Perform semantic segmentation within each anchor box from object detection to identify foreground (object) and background.

MEMORIAL UNIVERSITY

Text Classification using CNN

Import packages:

```
import collections
import pathlib
import datetime

import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras import losses
from tensorflow.keras import utils
from tensorflow.keras.layers import TextVectorization
```

MEMORIAL UNIVERSITY

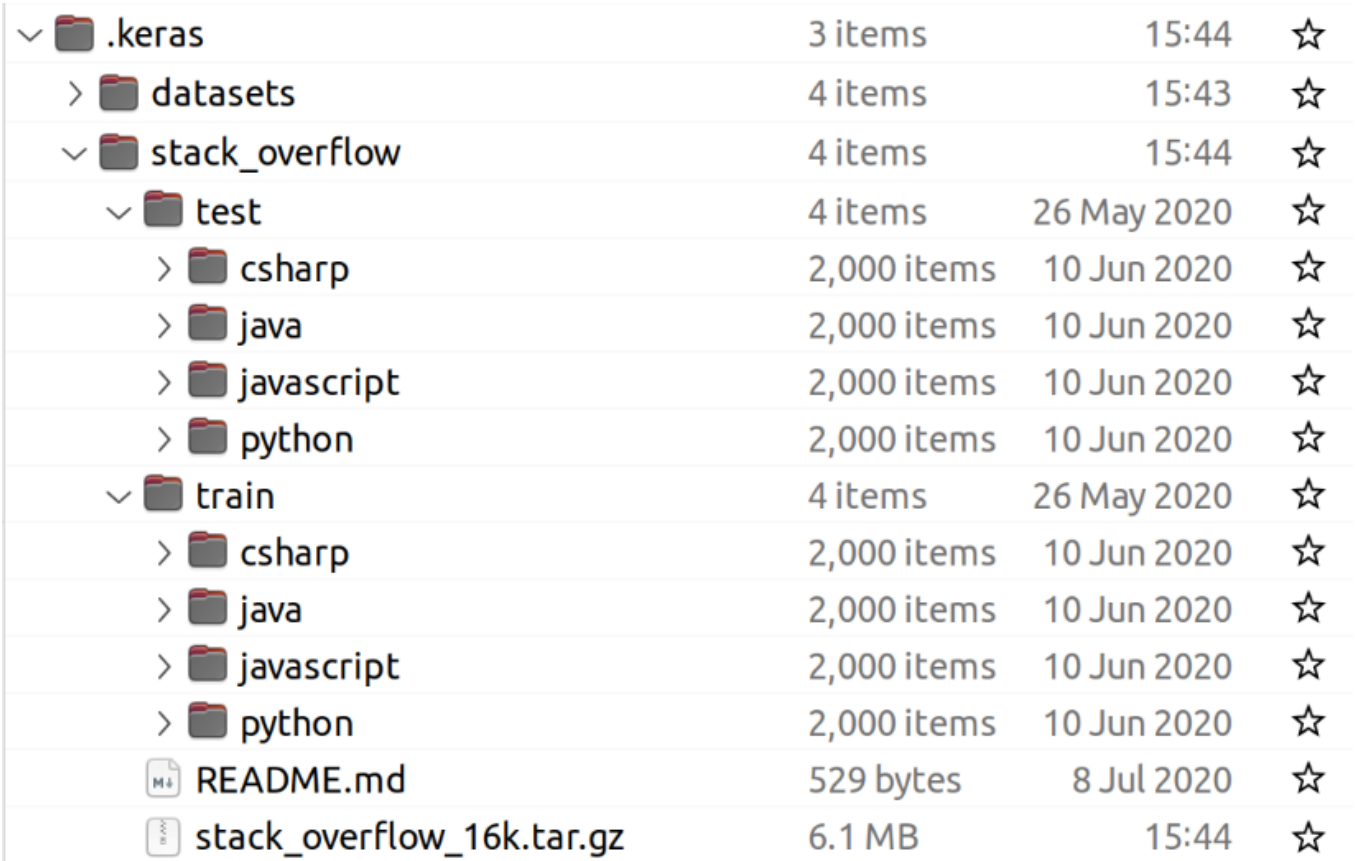
Text Classification using NN [cont'd]

Download the Stack Overflow dataset:

```
data_url = \
'http://download.tensorflow.org/data/stack_overflow_16k.tar.gz'

# Get the data
dataset_dir = utils.get_file(origin=data_url, untar=True,
                             cache_subdir='stack_overflow')

# Remember where we put it
dataset_dir = pathlib.Path(dataset_dir).parent
train_dir = dataset_dir/'train'
test_dir = dataset_dir/'test'

# Print a sample
sample_file = train_dir/'python/1755.txt'
with open(sample_file) as f:
    print(f.read())
```

MEMORIAL UNIVERSITY

Text Classification using NN [cont'd]



MEMORIAL UNIVERSITY

Text Classification using NN [cont'd]

Read the training data portion and split into training and validation set

```
raw_train_ds = \
utils.text_dataset_from_directory(
    train_dir, batch_size=32,
    validation_split=0.2,
    subset='training', seed=42)

raw_val_ds = \
utils.text_dataset_from_directory(
    train_dir, batch_size=32,
    validation_split=0.2,
    subset='validation', seed=42)
```

Read the test set portion of the dataset:

```
raw_test_ds = utils.text_dataset_from_directory(
    test_dir, batch_size=32)
```

MEMORIAL UNIVERSITY

Text Classification using NN [cont'd]

Pre-Processing

- **Standardization**: Remove punction, remove HTML elements, etc.
- **Tokenization**: Split strings to tokens (e.g. sentences into words on whitespace)
- **Vectorization**: Convert tokens into numbers for input into a neural network

Bag-of-Word Model

- Counts frequency of words/tokens
- Neglects order

```
John likes to watch movies. Mary likes movies too.
{"John":1,"likes":2,"to":1,"watch":1,
 "movies":2,"Mary":1,"too":1}
```

Text Classification Using NN [cont'd]

Use the Keras `TextVectorization` pre-processing layer:

```
multi_hot_vectorize_layer = TextVectorization(
    max_tokens=10000,
    standardize='lower_and_strip_punctuation',
    split='whitespace',
    output_mode='multi_hot')

# Get only the text from the training set
train_text = raw_train_ds.map(lambda text,labels: text)
multi_hot_vectorize_layer.adapt(train_text)
```

MEMORIAL UNIVERSITY

Text Classification Using NN [cont'd]

- The "multi-hot" vectorization layer outputs a single integer array per mini-batch, containing 1s in all elements where the token mapped to that index exists at least once in the mini-batch item.

```
# Retrieve a batch from the dataset
text_batch, label_batch = next(iter(raw_train_ds))

# Applying the text vectorization layer
# to the first example and print its output
print(text_batch[0])
print(list(multi_hot_vectorize_layer(text_batch[0]).numpy()))
```

MEMORIAL UNIVERSITY

Text Classification Using NN [cont'd]

Define a simple model, set loss function, optimizer and a metric to collect:

```
multi_hot_model = tf.keras.Sequential([
    multi_hot_vectorize_layer,
    layers.Dense(4)])

multi_hot_model.compile(
    loss= \
      losses.SparseCategoricalCrossentropy(from_logits=True),
    optimizer='adam',
    metrics=['accuracy'])
```

MEMORIAL UNIVERSITY

Text Classification Using NN [cont'd]

Train the model for 25 epochs, showing validation set accuracy after each epoch:

```
multi_hot_model.fit(raw_train_ds,
                    validation_data=raw_val_ds,
                    epochs=25)
```

MEMORIAL UNIVERSITY

Hands-On Exercises

Adapt the network architecture to identify the impact on training and validation performance of the following:

1. Vocabulary size (originally 10 000)
2. Number of layers (originally 1)

Comment on your findings and identify the best model.

MEMORIAL UNIVERSITY

Text Classification Using CNN [cont'd]

Use the Keras `TextVectorization` pre-processing layer, this time with integer outputs:

```
int_vectorize_layer = TextVectorization(
    max_tokens=10000,
    standardize='lower_and_strip_punctuation',
    split='whitespace',
    output_mode='int')

# Get only the text from the training set
train_text = raw_train_ds.map(lambda text,labels: text)
# Adapt the layer
int_vectorize_layer.adapt(train_text)
```

MEMORIAL UNIVERSITY

Text Classification Using CNN [cont'd]

- The "int" vectorization layer outputs integer indices, one integer index per split string token.

```
# Retrieve a batch from the dataset
text_batch, label_batch = next(iter(raw_train_ds))

# Applying the text vectorization layer
# to the first example and print its output
print(text_batch[0])
print(list(int_vectorize_layer(text_batch[0]).numpy()))
```

MEMORIAL UNIVERSITY

Word Embeddings

One-hot encoding

	cat	mat	on	sat	the
the =>	0	0	0	0	1
cat =>	1	0	0	0	0
sat =>	0	0	0	1	0
...	...				

A 4-dimensional embedding

cat =>	1.2	-0.1	4.3	3.2
mat =>	0.4	2.5	-0.9	0.5
on =>	2.1	0.3	0.1	0.4
...	...			

`https://www.tensorflow.org/text/guide/word_embeddings`

- Embedding matrix elements are **learnable parameters**
- Dimension reduction
- May be useful to measure similarity of terms

MEMORIAL UNIVERSITY

Text Classification Using NN [cont'd]

Define a 1D-CNN model, set loss function, optimizer and a metric to collect:

```
int_model = tf.keras.Sequential([
  int_vectorize_layer,
  layers.Embedding(10001, 64, mask_zero=True),
  layers.Dropout(0.5),
  layers.Conv1D(filters=64, kernel_size=5, \
      padding="valid", activation="relu", strides=2),
  layers.GlobalMaxPooling1D(),
  layers.Dense(4)
  ])

int_model.compile(
    loss= \
losses.SparseCategoricalCrossentropy(from_logits=True),
    optimizer='adam',
    metrics=['accuracy'])
```

MEMORIAL UNIVERSITY

Text Classification Using NN [cont'd]

Train the model for 25 epochs:

```
int_model.fit(raw_train_ds,
              validation_data=raw_val_ds,
              epochs=25)
```

MEMORIAL UNIVERSITY

Hands-On Exercises

Adapt the network architecture to identify the impact on training and validation performance of the following:

1. Vocabulary size (originally 10 000)
2. Embedding size (originally 64)
3. Dropout probability (originally 0.5)
4. Convolution kernel size (originally 5)
5. Number of convolution filters (originally 64)
6. Convolution stride (originally 2)
7. Number of 1D-Conv layers (originally 1)

Comment on your findings and identify the best model.

MEMORIAL UNIVERSITY