Transfer Learning and Embeddings

CS344
Deep Learning

WELLESLEY

Transfer Learning

Transfer learning is taking knowledge a NN has learned in one task and applying it to learning a different task

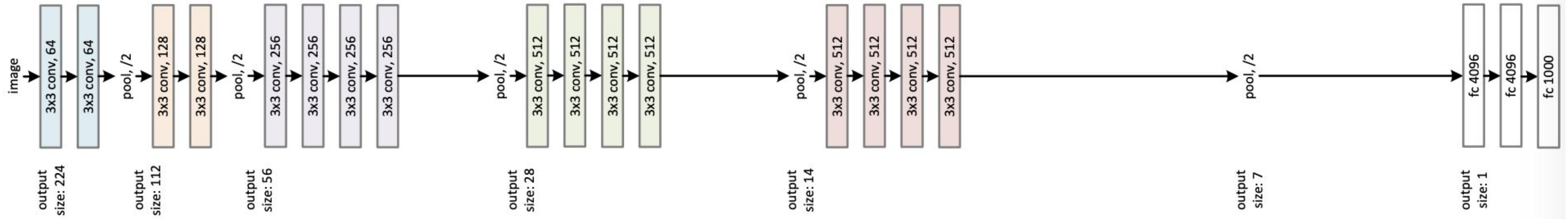
Transfer Learning

Suppose some NN has been trained on a very large set of images

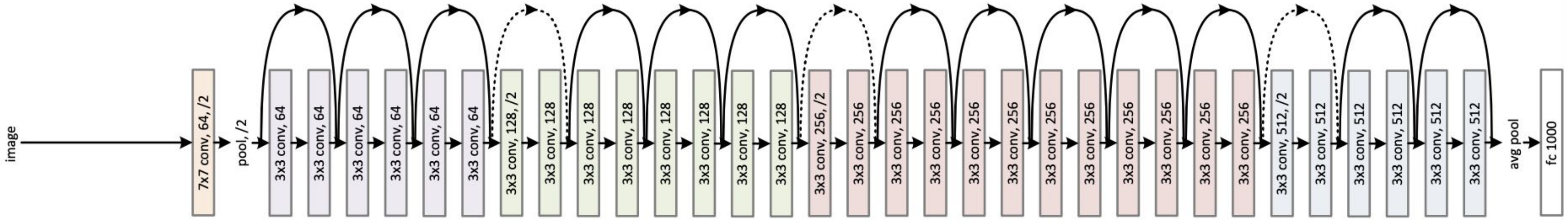
We can use this NN as a starting point for solving some other image problem

Transfer Learning

Trained on over 1M images belonging to 1000 classes

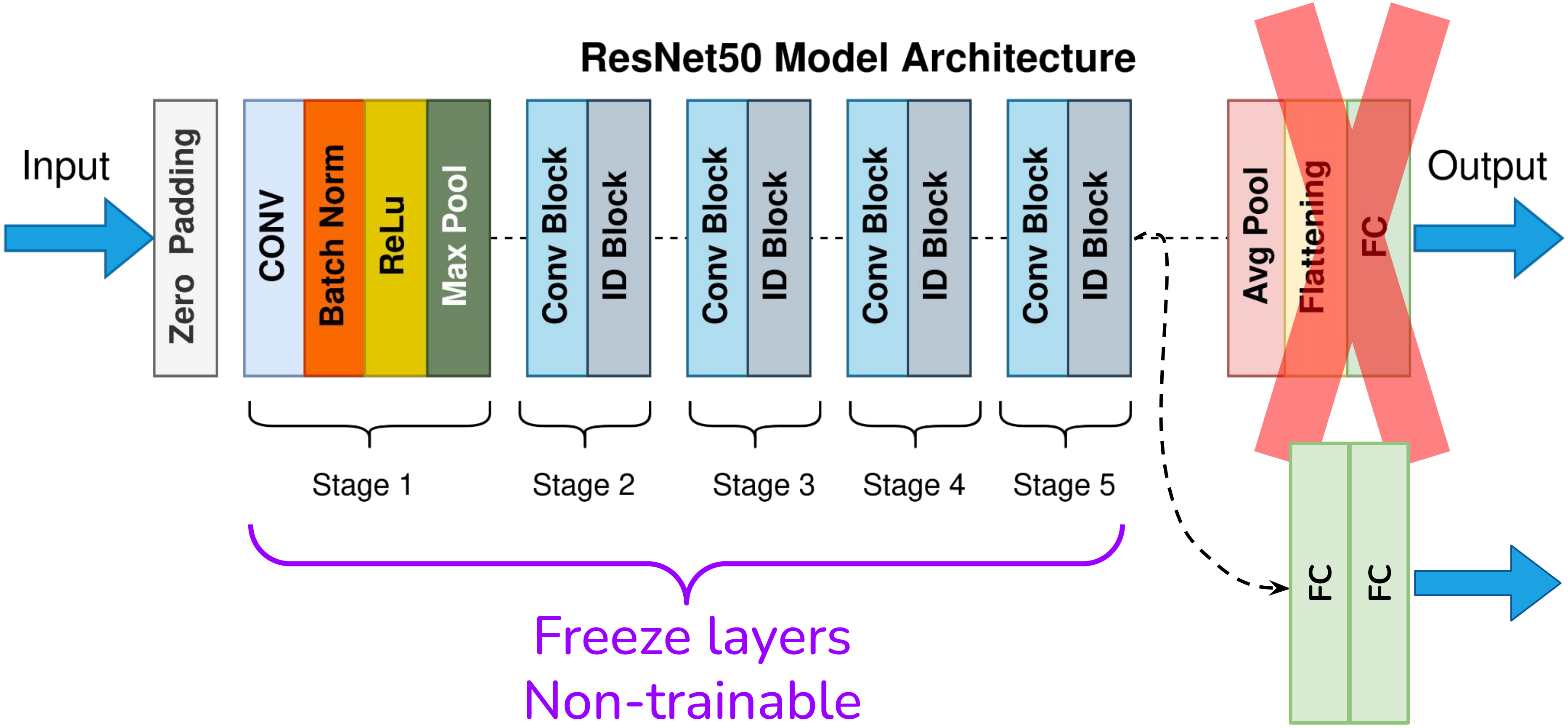


VGG-19



ResNet 34

ResNet 50


ResNet50 Model Architecture
Input
Zero Padding
CONV
Batch Norm
ReLu
Max Pool
Conv Block
ID Block
Conv Block
ID Block
Conv Block
ID Block
Conv Block
ID Block
Avg Pool
Flattening
FC
Output
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
FC
FC
Freeze layers
Non-trainable

ResNet 50

```
import tensorflow as tf
```

```
R50 = tf.keras.applications.ResNet50(pooling='avg', weights='imagenet')
```

ResNet 50

```
R50.summary()
```

```
 Layer (type)                Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3  0           []
                             )]

 conv1_pad (ZeroPadding2D)   (None, 230, 230, 3)  0           ['input_1[0][0]']

 conv1_conv (Conv2D)         (None, 112, 112, 64  9472        ['conv1_pad[0][0]']
                             )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256      ['conv1_conv[0][0]']
                             )

 conv1_relu (Activation)     (None, 112, 112, 64  0           ['conv1_bn[0][0]']
                             )

 pool1_pad (ZeroPadding2D)   (None, 114, 114, 64  0           ['conv1_relu[0][0]']

• • •

 conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5_block3_2_relu[0][0]']

 conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5_block3_3_conv[0][0]']
 ization)

 conv5_block3_add (Add)         (None, 7, 7, 2048)   0           ['conv5_block2_out[0][0]',
                                                                  'conv5_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5_block3_add[0][0]']

 avg_pool (GlobalAveragePooling  (None, 2048)        0           ['conv5_block3_out[0][0]']
 2D)

 predictions (Dense)            (None, 1000)         2049000     ['avg_pool[0][0]']

==================================================================================================
Total params: 25,636,712
Trainable params: 25,583,592
Non-trainable params: 53,120
```

ResNet 50

```
import tensorflow as tf
```

```
R50 = tf.keras.applications.ResNet50(pooling='avg', weights='imagenet')
```

```
R50 = tf.keras.applications.ResNet50(pooling='avg', weights='imagenet', include_top=False)
```

ResNet 50

```
R50.summary()
```

```
include_top=True
```

```
Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)           [(None, 224, 224, 3  0           []
                               )]

conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input_1[0][0]']

conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1_pad[0][0]']
                               )

conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1_conv[0][0]']
                               )

conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1_bn[0][0]']
                               )

pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1_relu[0][0]']
```

• • •

```
conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5_block3_2_relu[0][0]']

conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5_block3_3_conv[0][0]']
ization)

conv5_block3_add (Add)         (None, 7, 7, 2048)   0           ['conv5_block2_out[0][0]',
                                                                 'conv5_block3_3_bn[0][0]']

conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5_block3_add[0][0]']

avg_pool (GlobalAveragePooling  (None, 2048)        0           ['conv5_block3_out[0][0]']
2D)

predictions (Dense)            (None, 1000)         2049000     ['avg_pool[0][0]']

==================================================================================================
Total params: 25,636,712
Trainable params: 25,583,592
Non-trainable params: 53,120
```

ResNet 50

```
R50.summary()
```

```
include_top=False
```

```
______________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==============================================================================================
 input_1 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input_1[0][0]']

 conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1_relu[0][0]']

                                  • • •

 conv5_block3_3_bn (BatchNormal  (None, None, None,   8192       ['conv5_block3_3_conv[0][0]']
 ization)                       2048)

 conv5_block3_add (Add)         (None, None, None,   0           ['conv5_block2_out[0][0]',
                                2048)                             'conv5_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, None, None,   0           ['conv5_block3_add[0][0]']
                                2048)

 avg_pool (GlobalAveragePooling  (None, 2048)        0           ['conv5_block3_out[0][0]']
 2D)

==============================================================================================
Total params: 23,587,712
Trainable params: 23,534,592
Non-trainable params: 53,120
```

ResNet 50

```
# Create model
model_R50 = tf.keras.Sequential([
    tf.keras.applications.ResNet50(include_top=False, pooling='avg', weights='imagenet'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(4, activation='softmax')
])

model_R50.layers[0].trainable = False
```

ResNet 50

```
# Create model
model_R50 = tf.keras.Sequential([
    tf.keras.applications.ResNet50(include_top=False, pooling='avg', weights='imagenet'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(4, activation='softmax')
])

model_R50.layers[0].trainable = False
```

```
model_R50.summary()
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 dense (Dense)               (None, 128)               262272

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 23,850,500
Trainable params: 262,788
Non-trainable params: 23,587,712
```

ResNet 50

```
# Create model
model_R50 = tf.keras.Sequential([
    tf.keras.applications.ResNet50(include_top=False, pooling='avg', weights='imagenet'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(4, activation='softmax')
])

model_R50.layers[0].trainable = False
```

```
# Compile and train model
model_R50.compile(
  optimizer='adam',
  loss=tf.keras.losses.SparseCategoricalCrossentropy(),
  metrics=['accuracy'])

model_R50.fit(train_ds, validation_data=val_ds, epochs=5)
```

Transfer Learning

Why not take advantage?

Transfer learning is taking knowledge a NN has learned in one task and applying it to learning a different task

Many NNs have been trained on massive datasets for long periods of time, with their architectures and hyperparameters tuned

These pre-trained NNs have already learned low-level (and mid-level and high-level) features from a very large dataset

Embeddings

Embedding

An ***embedding*** is a way to represent high-dimensional data in a low-dimensional space in a way that captures some of the structure, similarity, or semantic meaning of the data

Non-Embedding

In English, words are represented by a sequence of letters, but the letters have no relationship to the meaning of the word

opt top pot

cat cap car

What if all words that started with the letter “a” pertained to animals, and all words that started with the letter “b” were verbs?

Embedding

An ***embedding*** is a way to represent high-dimensional data in a low-dimensional space in a way that captures some of the structure, similarity, or semantic meaning of the data

In an ***embedding*** in machine learning, data are normally represented by a vector (1D array) of numbers, where each position in the vector corresponds to a feature

ResNet 50

`include_top=False`

As input, start with (224, 224, 3) image

50,176 pixels
150,528 features

```
Layer (type)                   Output Shape         Param #     Connected to
==================================================================================
input_1 (InputLayer)           [(None, 224, 224, 3  0           []
                               )]

conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input_1[0][0]']

conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1_pad[0][0]']
                               )

conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1_conv[0][0]']
                               )
```

Embedding is lower-dimensional representation that captures some meaning in image, though 2,048 features may not be interpretable

```
conv5_block3_add (Add)         (None, None, None,   0           ['conv5_block2_out[0][0]',
                               2048)                             'conv5_block3_3_bn[0][0]']

conv5_block3_out (Activation)  (None, None, None,   0           ['conv5_block3_add[0][0]']
                               2048)

avg_pool (GlobalAveragePooling  (None, 2048)        0           ['conv5_block3_out[0][0]']
2D)

==================================================================================
Total params: 23,587,712
Trainable params: 23,534,592
Non-trainable params: 53,120
```

2,048 embedding

Image Embedding

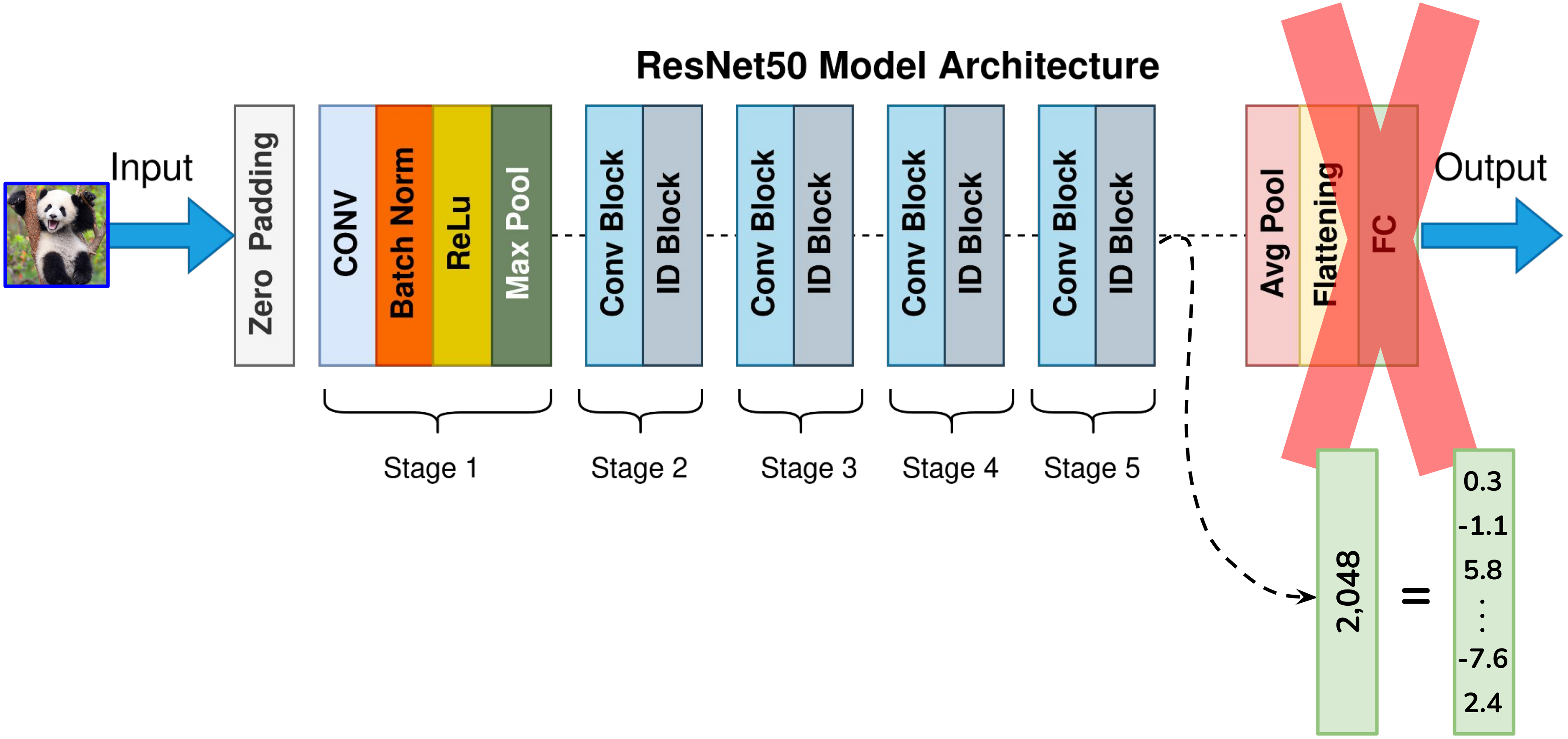


Image Embedding

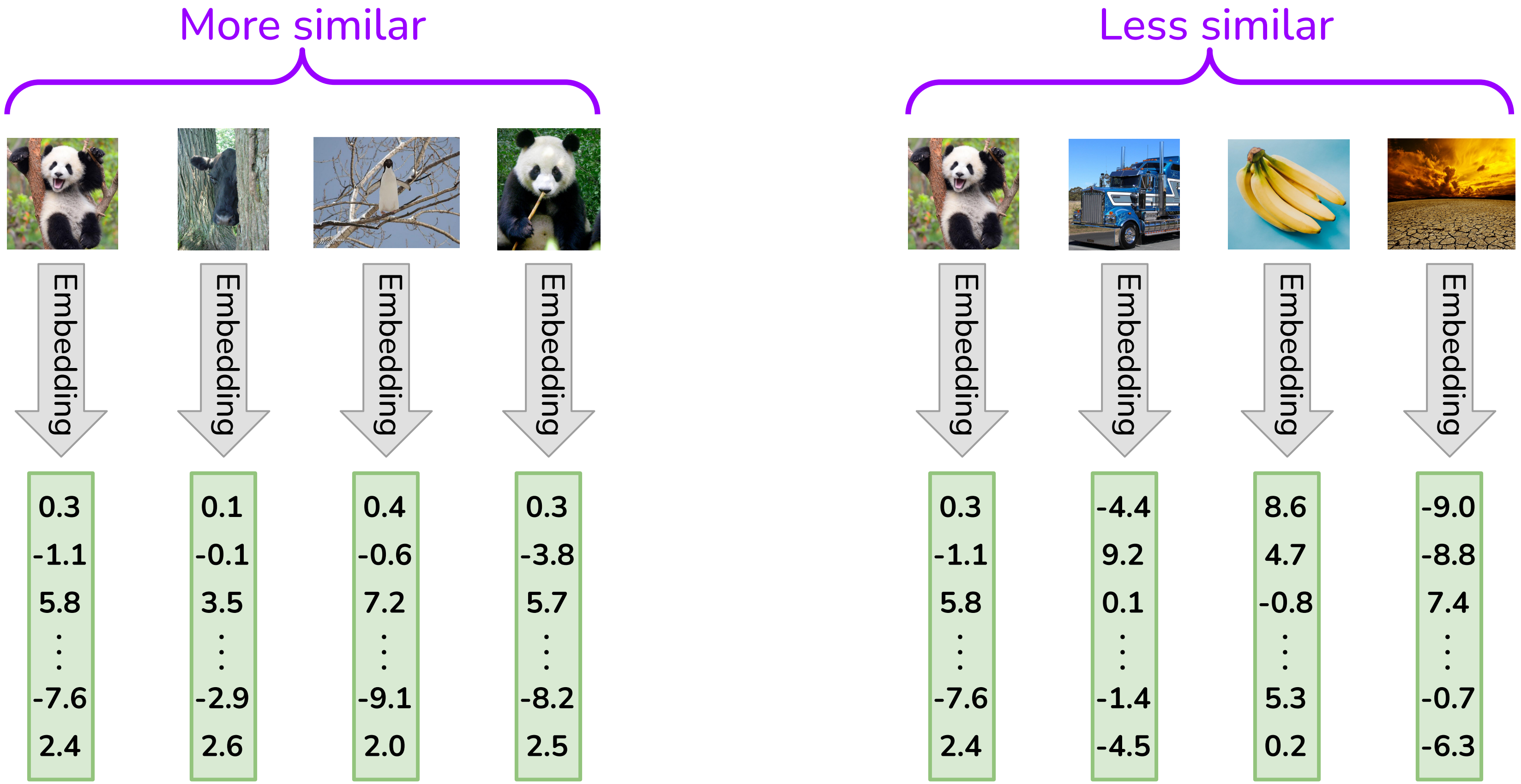
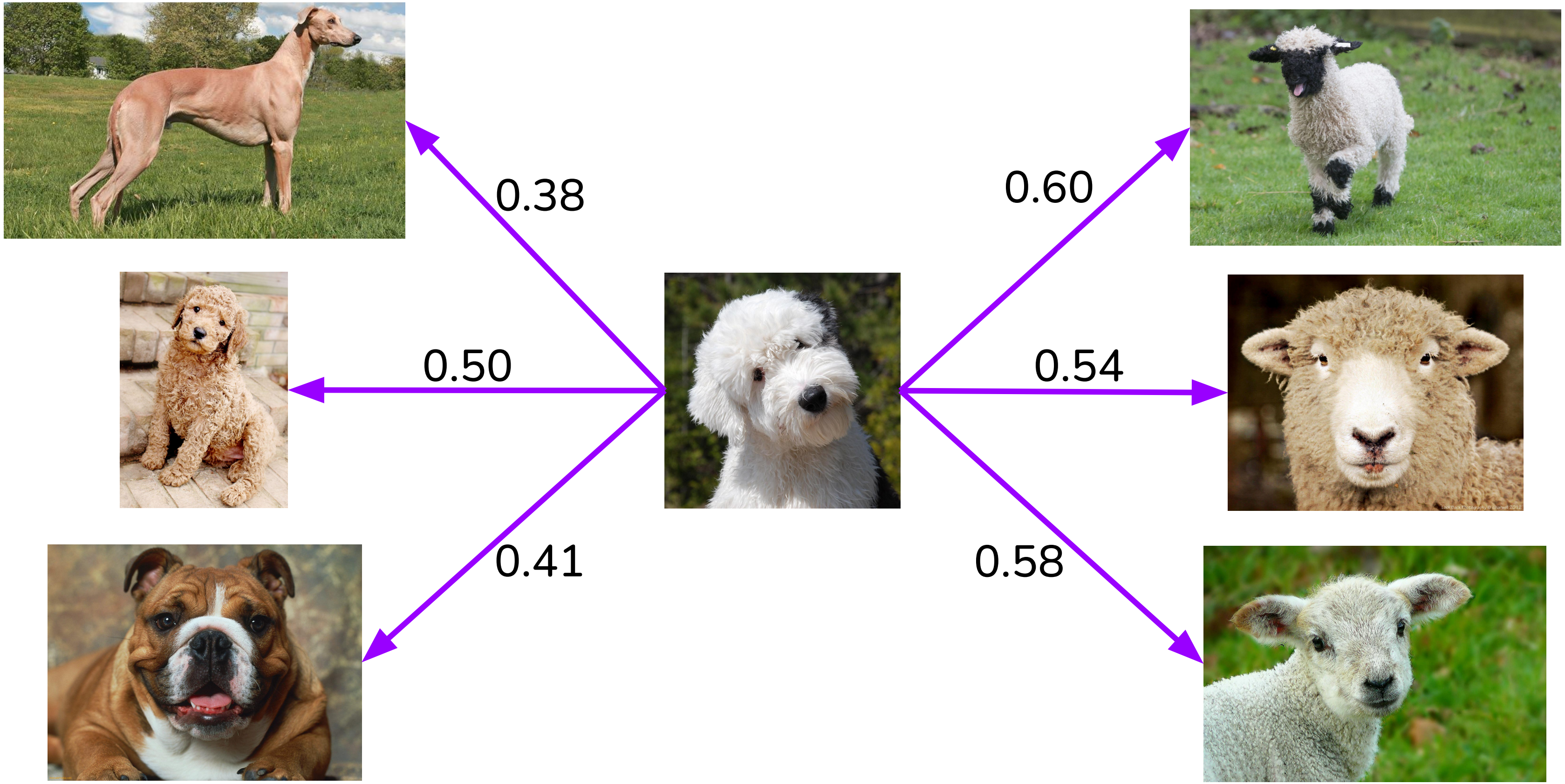


Image Similarity (Cosine)



Word Encoding

Apple	College	Ruby	Studying	Fox	Pi
1	2	3	4	5	6

Word Encoding

Apple	College	Ruby	Studying	Fox	Pi
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	0	0	0

Word Embedding

	Apple	College	Ruby	Studying	Fox	Pi
Size	-0.5	0.6	-0.8	0.1	0.3	-0.6
Red	0.8	0.03	0.91	0.0	0.7	0.01
Verb	0.01	-0.01	-0.07	0.99	0.4	0.0
Scholastic	0.2	0.97	0.03	0.87	0.02	0.3
Animal	0.05	0.01	-0.04	-0.02	0.99	-0.1
Numerical	-0.02	0.21	0.0	0.3	0.01	0.99
Cost	-0.8	0.92	0.94	0.2	0.04	0.06

GloVe (Global Vectors for word representation)

Transfer learning!

Trained on 6 billion words of text
from Wikipedia and from news stories

Each word is represented by 50 dimensional vector

The 50 features may not be interpretable

GloVe Embedding

50 dimensional

Apple	College	Ruby	Studying	Fox	Pi
0.52	-1.23	0.16	0.29	0.44	0.4
-0.83	1.42	0.91	0.35	0.06	1.07
0.5	-0.69	-0.55	-0.87	0.16	0.44
1.29	-1.16	1.39	-0.73	0.93	0.64
0.12	0.0	-0.14	-0.08	0.19	0.33
⋮	⋮	⋮	⋮	⋮	⋮
0.27	0.32	-0.25	-0.11	1.51	0.15

Word Similarity (Cosine)

0.87 College University

0.57 Fox Wolf

0.40 Apple Red

0.09 Fox Pi

Apple	College	Ruby	Studying	Fox	Pi
0.52	-1.23	0.16	0.29	0.44	0.4
-0.83	1.42	0.91	0.35	0.06	1.07
0.5	-0.69	-0.55	-0.87	0.16	0.44
1.29	-1.16	1.39	-0.73	0.93	0.64
0.12	0.0	-0.14	-0.08	0.19	0.33
⋮	⋮	⋮	⋮	⋮	⋮
0.27	0.32	-0.25	-0.11	1.51	0.15

Nearest Neighbors

0.92	Truck
0.89	Cars
0.88	Vehicle
0.85	Driver
0.84	Driving
0.82	Bus
0.82	Vehicles
0.79	Parked
0.79	Motorcycle
0.78	Taxi

What are the k=10 words most similar to “car”?

t-SNE (t-distributed Stochastic Neighbor Embedding)

How can we visualize 50-dimensional data?

Embed it in 2 dimensions. And plot it!

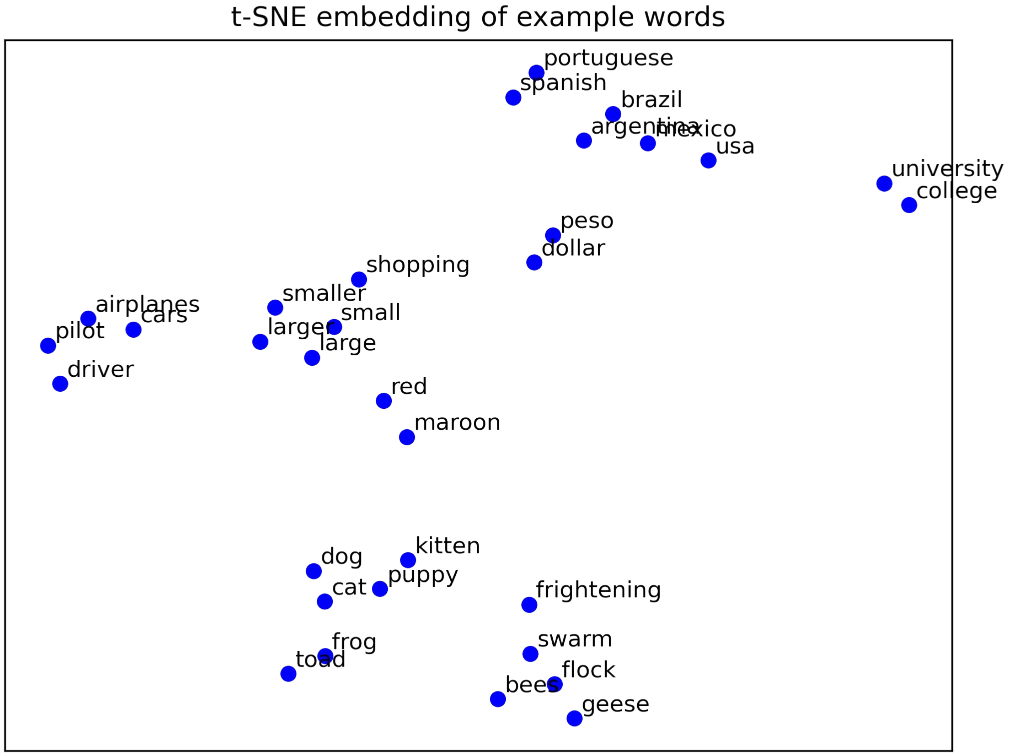
Apple	College	Ruby	Studying	Fox	Pi
0.52	-1.23	0.16	0.29	0.44	0.4
-0.83	1.42	0.91	0.35	0.06	1.07
0.5	-0.69	-0.55	-0.87	0.16	0.44
1.29	-1.16	1.39	-0.73	0.93	0.64
0.12	0.0	-0.14	-0.08	0.19	0.33
⋮	⋮	⋮	⋮	⋮	⋮
0.27	0.32	-0.25	-0.11	1.51	0.15

PCA is a linear reduction, whereas t-SNE is non-linear.

t-SNE (t-distributed Stochastic Neighbor Embedding)

How can we visualize 50-dimensional data?

Embed it in 2 dimensions. And plot it!

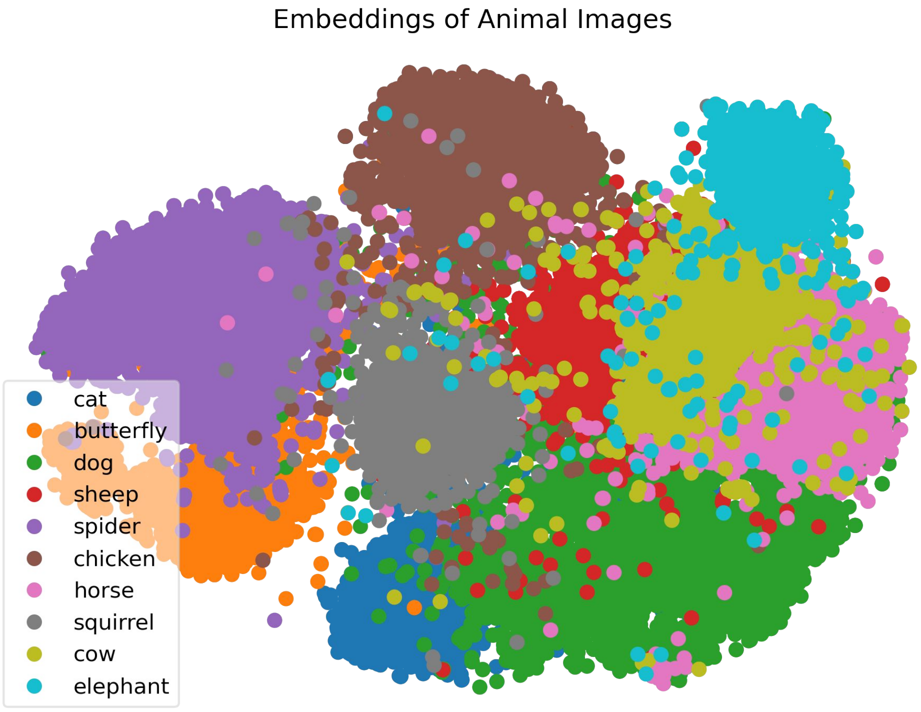


PCA is a linear reduction, whereas t-SNE is non-linear.

t-SNE (t-distributed Stochastic Neighbor Embedding)

How can we visualize 2048-dimensional data?

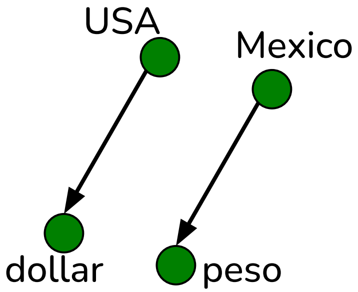
Embed it in 2 dimensions. And plot it!



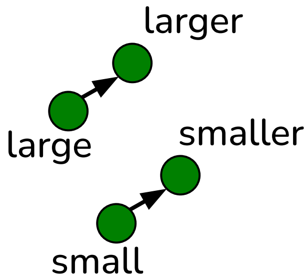
Dataset of 26k+ images of 10 different animals

Word Analogies

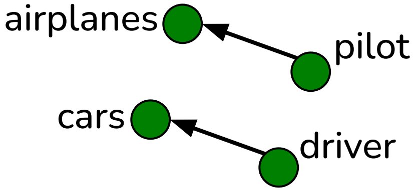
USA is to dollar as Mexico is to _________



small is to smaller as large is to _________



pilot is to airplanes as driver is to _________



Word Analogies

USA is to dollar as Mexico is to _________

A is to *B* as *C* is to *D*

For every word *D*, calculate the similarity between *A*-*B* and *C*-*D*, and choose the word *D* that yields the highest similarity