Transfer Learning Theory and it's Applications using Tensorflow & Keras

Babak Badnava

PyCon 2018

Traditional ML

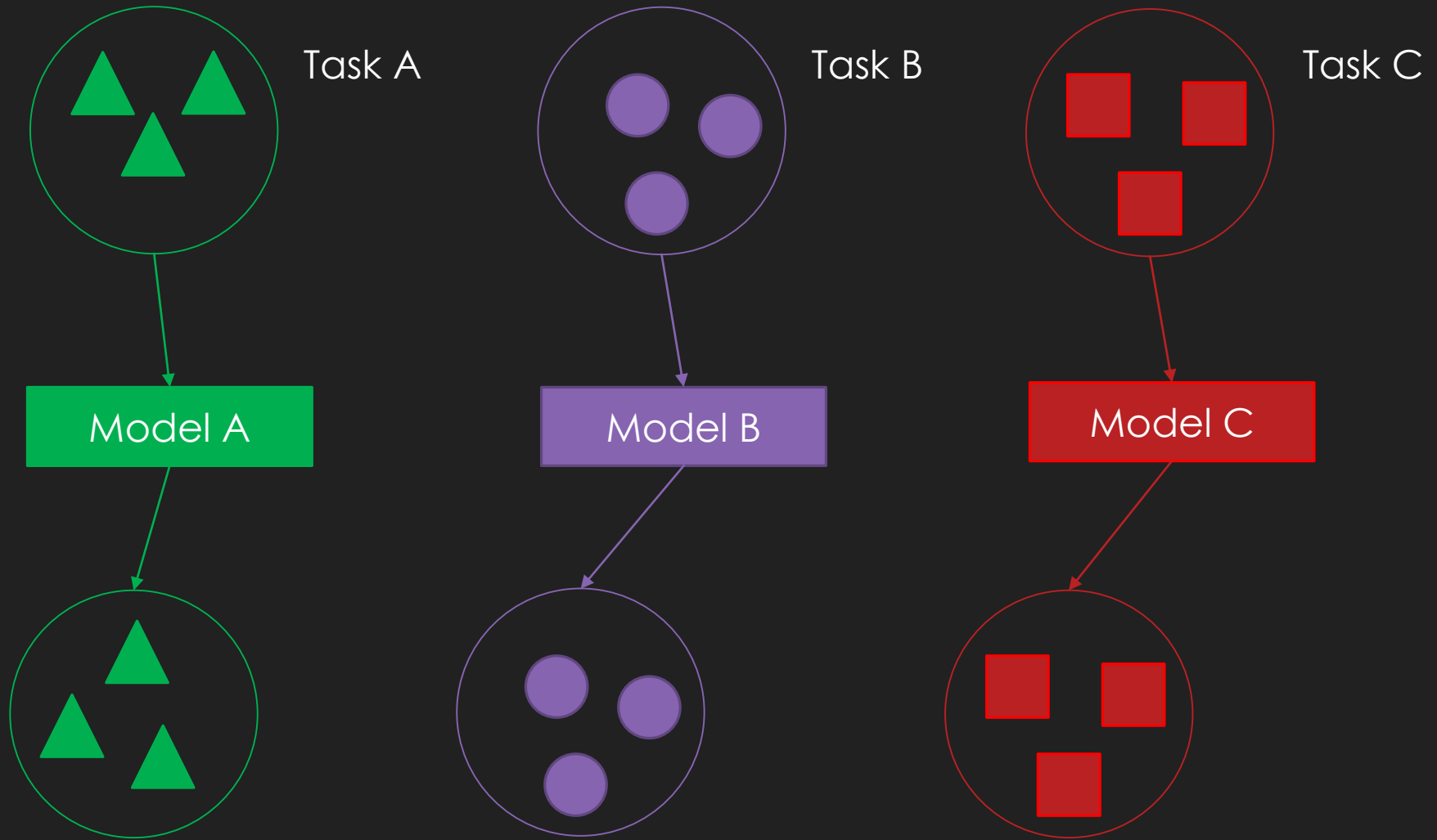
Problem ?

Solution ?

Approaches

TL in Practice

Final Part



Problem Definition!

Problem ?

Solution ?

Approaches

TL in Practice

Final Part

- Complex problems
- Insufficient labeled data
- Long training time

Solution ?

- Assumptions ?
  - Related data for different task
  - Previously learned tasks
- Solution ?
  - Use them

Transfer Learning framework

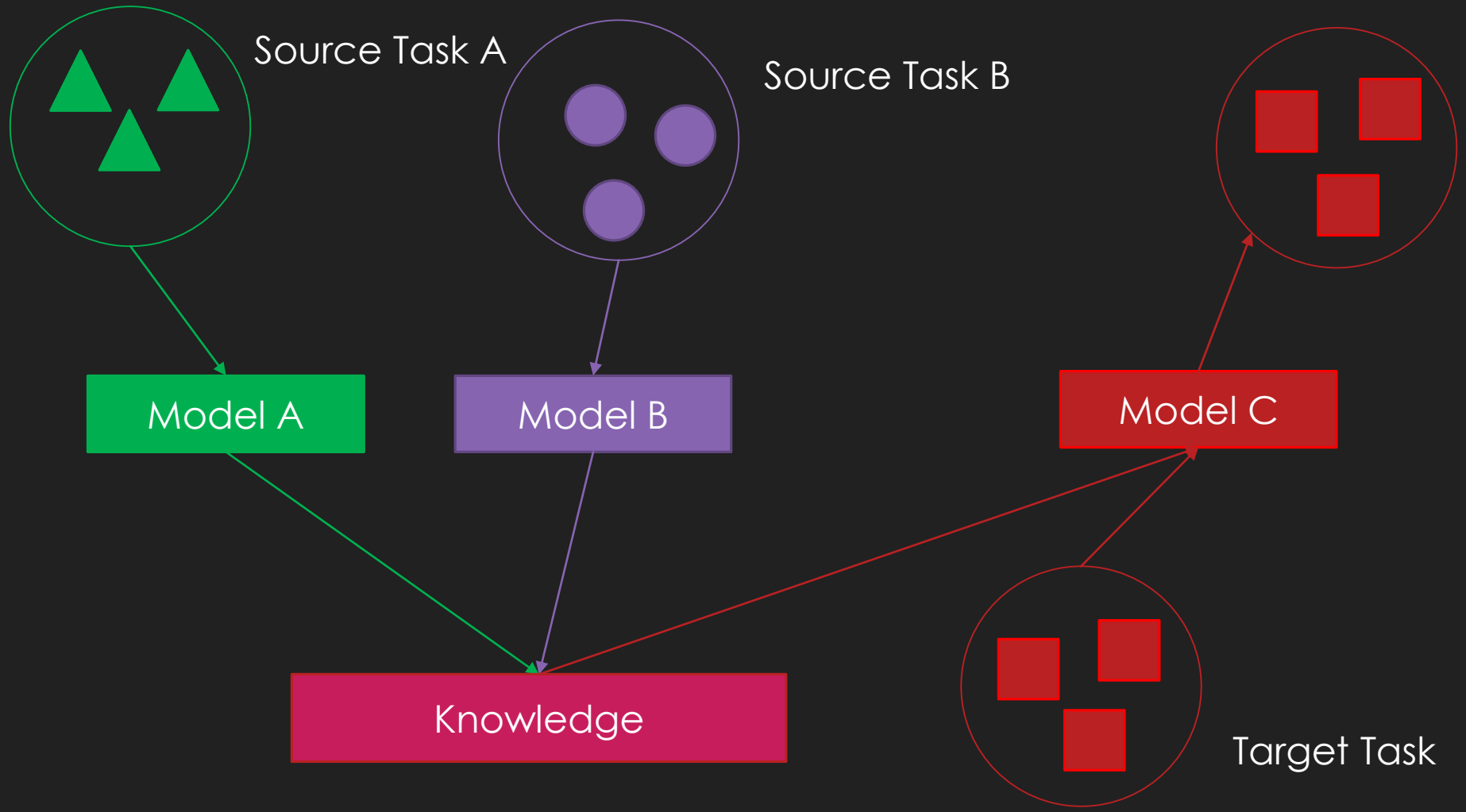
Problem ?

Solution ?

Approaches

TL in Practice

Final Part



Transfer Learning and Neural Networks

- Problem ?
- Solution ?
- Approaches
- TL in Practice
- Final Part

- Pre-trained models
  - Trains with large dataset
  - Use for specific task
- Fixed feature extractor
  - Autoencoders
  - Freezing pre-trained model
- Fine-tuning

Pre-trained models on ImageNet in Keras

Problem ?

Solution ?

Approaches

TL in Practice

Final Part

- Xception
- VGG16
- VGG19
- ResNet50
- InceptionV3
- ……

https://keras.io/applications/

ResNet50 for specific task

- Problem ?
- Solution ?
- Approaches
- TL in Practice
- Final Part

```python
from keras.applications.resnet50 import ResNet50
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

model = ResNet50(weights='imagenet')

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
```

Extract features with VGG16

- Problem ?
- Solution ?
- Approaches
- TL in Practice
- Final Part

```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
import numpy as np

model = VGG16(weights='imagenet', include_top=False)

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

features = model.predict(x)
```

Extract features from an intermediate layer with VGG19

- Problem ?
- Solution ?
- Approaches
- TL in Practice
- Final Part

```
from keras.applications.vgg19 import VGG19
from keras.preprocessing import image
from keras.applications.vgg19 import preprocess_input
from keras.models import Model
import numpy as np

base_model = VGG19(weights='imagenet')
model = Model(inputs=base_model.input, outputs=base_model.get_layer('block4_pool').output)

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

block4_pool_features = model.predict(x)
```

Fine-tune InceptionV3 on a new set of classes

- Problem ?
- Solution ?
- Approaches
- TL in Practice
- Final Part

```python
from keras.applications.inception_v3 import InceptionV3
from keras.preprocessing import image
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D
from keras import backend as K

# create the base pre-trained model
base_model = InceptionV3(weights='imagenet', include_top=False)

# add a global spatial average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
# let's add a fully-connected layer
x = Dense(1024, activation='relu')(x)
# and a logistic layer -- let's say we have 200 classes
predictions = Dense(200, activation='softmax')(x)


model = Model(inputs=base_model.input, outputs=predictions)
```

Fine-tune InceptionV3 on a new set of classes(2)

- Problem ?
- Solution ?
- Approaches
- TL in Practice
- Final Part

```
for layer in base_model.layers:
    layer.trainable = False

model.compile(optimizer='rmsprop', loss='categorical_crossentropy')

model.fit_generator(...)

for layer in model.layers[:249]:
   layer.trainable = False
for layer in model.layers[249:]:
   layer.trainable = True

from keras.optimizers import SGD
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy')

model.fit_generator(...)
```

TensorFlow-Slim

Problem ?
Solution ?
Approaches
TL in Practice
Final Part

- lightweight high-level API of TensorFlow
- Many pre-trained models such as:
  - Inception
  - ResNet V1 50
  - ResNet V1 101
  - VGG 16

https://github.com/tensorflow/models/tree/master/research/slim

Pre-trained models

- Problem ?
- Solution ?
- Approaches
- TL in Practice
- Final Part

```python
import tensorflow as tf
import tensorflow.contrib.slim.nets as nets

slim = tf.contrib.slim
vgg = nets.vgg
train_log_dir = ...
if not tf.gfile.Exists(train_log_dir):
  tf.gfile.MakeDirs(train_log_dir)

with tf.Graph().as_default():
  # Set up the data loading:
  images, labels = ...
  # Define the model:
  predictions = vgg.vgg_16(images, is_training=True)
  # Specify the loss function:
  slim.losses.softmax_cross_entropy(predictions, labels)
  total_loss = slim.losses.get_total_loss()
  tf.summary.scalar('losses/total_loss', total_loss)
  # Specify the optimization scheme:
  optimizer = tf.train.GradientDescentOptimizer(learning_rate=.001)
  train_tensor = slim.learning.create_train_op(total_loss, optimizer)

  # Actually runs training.
  slim.learning.train(train_tensor, train_log_dir)
```

Transfer a specific layer

- Problem ?
- Solution ?
- Approaches
- TL in Practice
- Final Part

```
import tensorflow as tf

exclude = ['InceptionResnetV2/Logits', 'InceptionResnetV2/AuxLogits']
variables_to_restore = tf.contrib.slim.get_variables_to_restore(exclude = exclude)
saver = tf.train.Saver(variables_to_restore)
checkpoint_file = './inception_resnet_v2_2016_08_30.ckpt'
saver.restore(sess, checkpoint_file)
```

Freeze a layer

Problem ?

Solution ?

Approaches

TL in Practice

Final Part

```python
def dense_layer(self, input, out_dim, name, func=tf.nn.relu):
        in_dim = input.get_shape().as_list()[-1]
        d = 1.0 / np.sqrt(in_dim)
        with tf.variable_scope(name):
            w_init = tf.random_uniform_initializer(-d, d)
            b_init = tf.random_uniform_initializer(-d, d)
            w = tf.get_variable('w', dtype=tf.float32, shape=[in_dim, out_dim], initializer=w_init, trainable=False)
            b = tf.get_variable('b', shape=[out_dim], initializer=b_init, trainable=False)

            output = tf.matmul(input, w) + b
            if func is not None:
                output = func(output)

        return output
```

Freeze a layer

- Problem ?
- Solution ?
- Approaches
- TL in Practice
- Final Part

```python
def conv2d_layer(self, input, filter_size, out_dim, name, strides, func=tf.nn.relu):
        in_dim = input.get_shape().as_list()[-1]
        d = 1.0 / np.sqrt(filter_size * filter_size * in_dim)
        with tf.variable_scope(name):
            w_init = tf.random_uniform_initializer(-d, d)
            b_init = tf.random_uniform_initializer(-d, d)
            w = tf.get_variable('w',
                                shape=[filter_size, filter_size, in_dim, out_dim],
                                dtype=tf.float32,
                                initializer=w_init, trainable=False)
            b = tf.get_variable('b', shape=[out_dim], initializer=b_init, trainable=False)

            output = tf.nn.conv2d(input, w, strides=strides, padding='SAME') + b
            if func is not None:
                output = func(output)

        return output
```

Applications

Problem ?

Solution ?

Approaches

TL in Practice

Final Part

- Learning from simulations
  - Autonomous cars
  - Robotics
- Adapting to new domains
- Semi supervised learning

When and which?

Problem ?
Solution ?
Approaches
TL in Practice
Final Part

	Similar dataset	Different dataset
Small dataset	**Highest level features + classifier**	**Lower level features + classifier**
Large dataset	**Fine tune**	**Fine tune**

Thanks for attention.

Questions?