

Report 3

Transfer Learning

Vibhav Joshi

22th May, 2020

Link to google colab:

https://colab.research.google.com/drive/1--lxZ2OJ0fnsBUdD6_LweIQkWC5-BevS?usp=sharing



Transfer Learning


Pre-Training

When we train the network on a **large dataset(for example: ImageNet)** , we train all the parameters of the neural network and therefore the model is learned. It may take hours on your GPU.

Fine Tuning

We can give the new dataset to fine tune the pre-trained CNN.

Consider that the new dataset is almost similar to the original dataset used for pre-training. Since the new dataset is similar, the same weights can be used for extracting the features from the new dataset.

- 
1. If the new dataset is very small, it's better to train only the final layers of the network to avoid overfitting, keeping all other layers fixed. So remove the final layers of the pre-trained network. Add new layers . **Retrain only the new layers.**
 2. **If the new dataset is very much large, retrain the whole network** with initial weights from the pretrained model.

How to fine tune if the new dataset is very different from the original dataset ?

The earlier features of a ConvNet contain more **generic features (e.g. edge detectors or color blob detectors)**, but later layers of the ConvNet becomes progressively more specific to the details of the **classes contained in the original dataset.**



The earlier layers can help to extract the features of the new data.

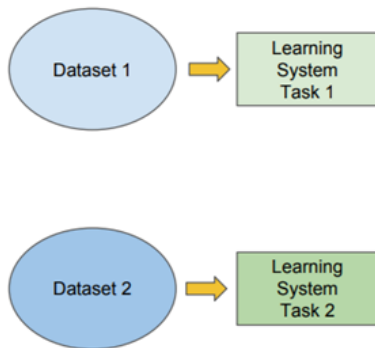
So it will be good if you fix the earlier layers and retrain the rest of the layers, if you got only small amount of data.

Traditional ML

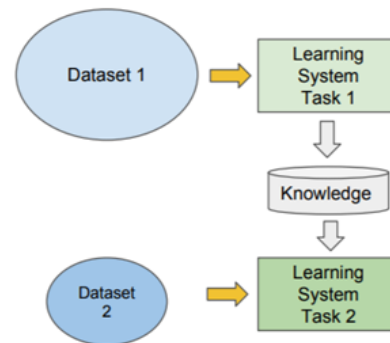
vs

Transfer Learning

- Isolated, single task learning:
 - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



- Learning of a new tasks relies on the previous learned tasks:
 - Learning process can be faster, more accurate and/or need less training data



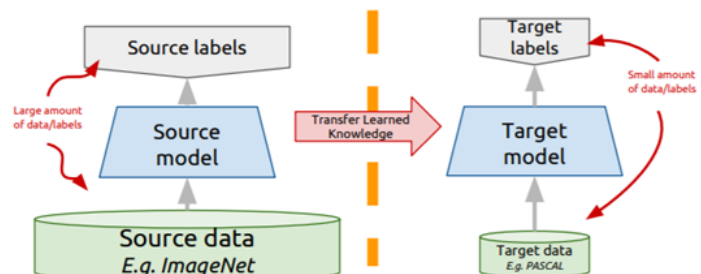
Transfer learning: idea

Instead of training a deep network from scratch for your task:

- Take a network trained on a different domain for a different **source task**
- Adapt it for your domain and your **target task**

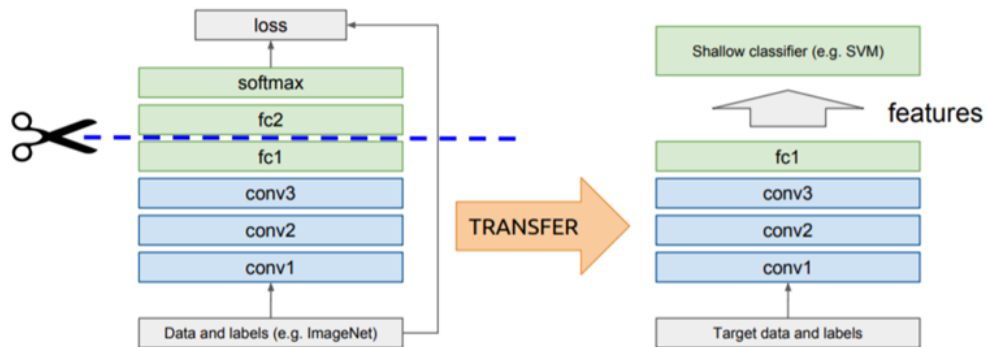
Variations:

- Same domain, different task
- Different domain, same task



Idea: use outputs of one or more layers of a network trained on a different task as generic feature detectors. Train a new shallow model on these features.

Assumes that $D_S = D_T$



My Project

- Researched about basics of Transfer Learning
- Learnt how different Architectures are used as Transfer Learning
- Used Transfer Learning and used VGG16 for training a dataset

```
IMAGE_SIZE = [224, 224]
```

```
train_path = '/content/data/Dataset/Train'
```

```
valid_path = '/content/data/Dataset/Val'
```

```
# add preprocessing layer to the front of VGG
```

```
vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet',  
include_top=False)
```

```
# don't train existing weights
```

```
for layer in vgg.layers:
```

```
    layer.trainable = False
```

```
# useful for getting number of classes

folders = glob('/content/data/Dataset/Train/*')


# our layers - you can add more if you want
x = Flatten()(vgg.output)

# x = Dense(1000, activation='relu')(x)

prediction = Dense(len(folders), activation='softmax')(x)


# create a model object
model = Model(inputs=vgg.input, outputs=prediction)


# view the structure of the model
model.summary()


# tell the model what cost and optimization method to use
model.compile(

    loss='categorical_crossentropy',

    optimizer='adam',

    metrics=['accuracy'])
```



```
)
```

```
from keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(rescale = 1./255,  
                                   shear_range = 0.2,  
                                   zoom_range = 0.2,  
                                   horizontal_flip = True)
```

```
test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
training_set =  
train_datagen.flow_from_directory('/content/data/Dataset/Train',  
                                  target_size = (224, 224),  
                                  batch_size = 32,  
                                  class_mode =  
'categorical')
```

```
test_set = test_datagen.flow_from_directory('/content/data/Dataset/Val',  
                                             target_size = (224, 224),  
                                             batch_size = 32,  
                                             class_mode = 'categorical')
```

—