Recap: Transfer Learning

MLP and CNN

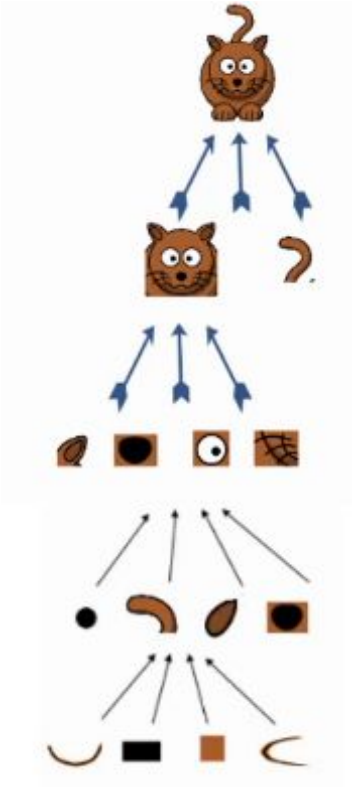


Image feature

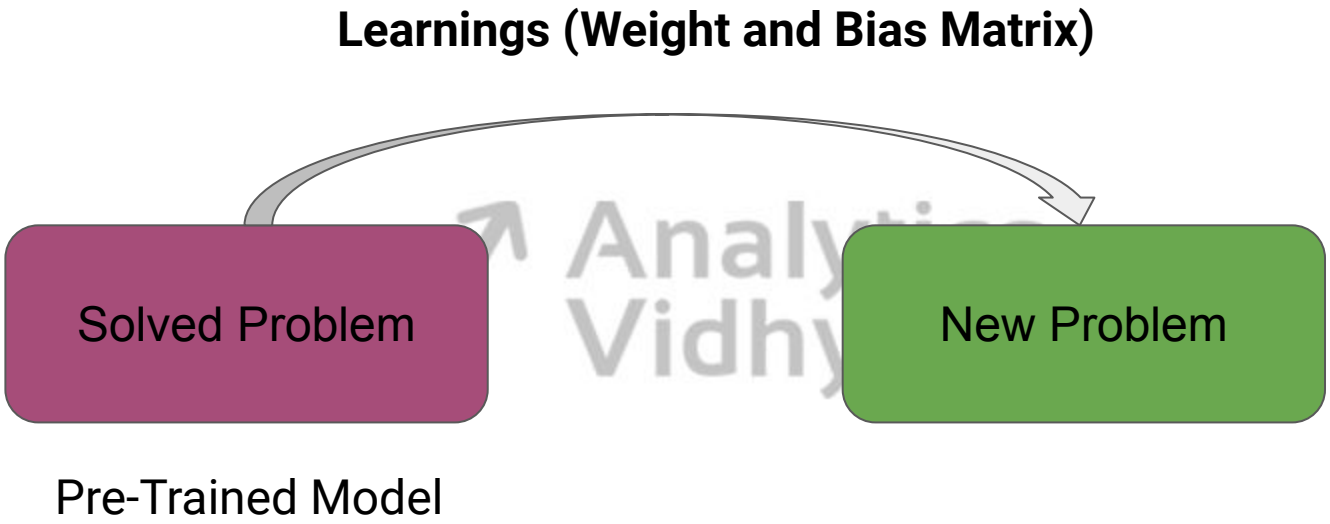
Shapes

Edges

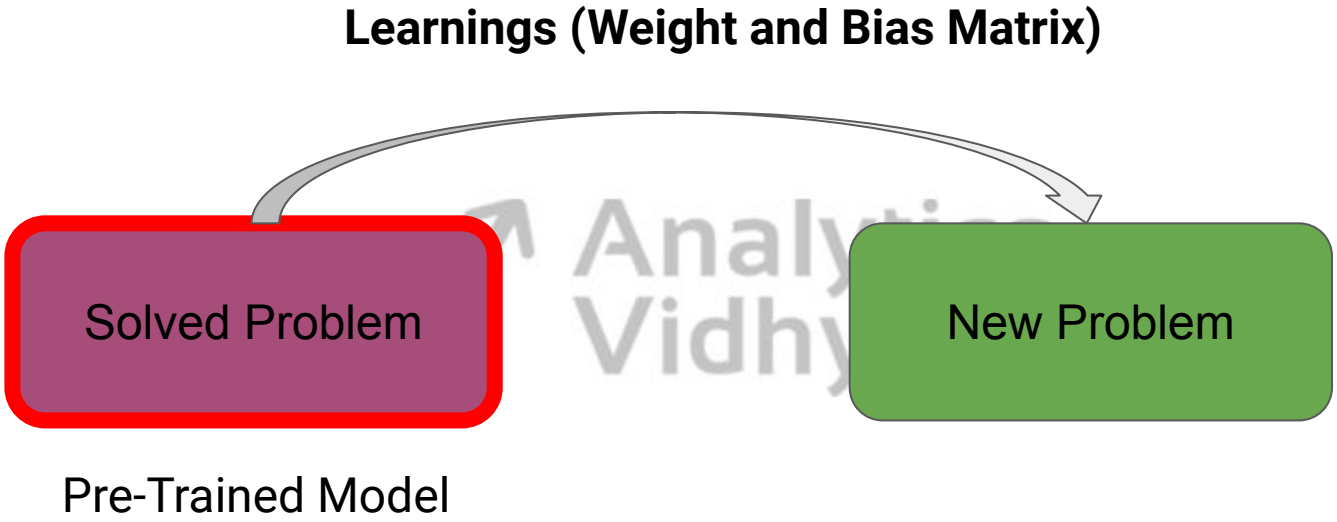
High Dimensional Features

Low Dimensional Features

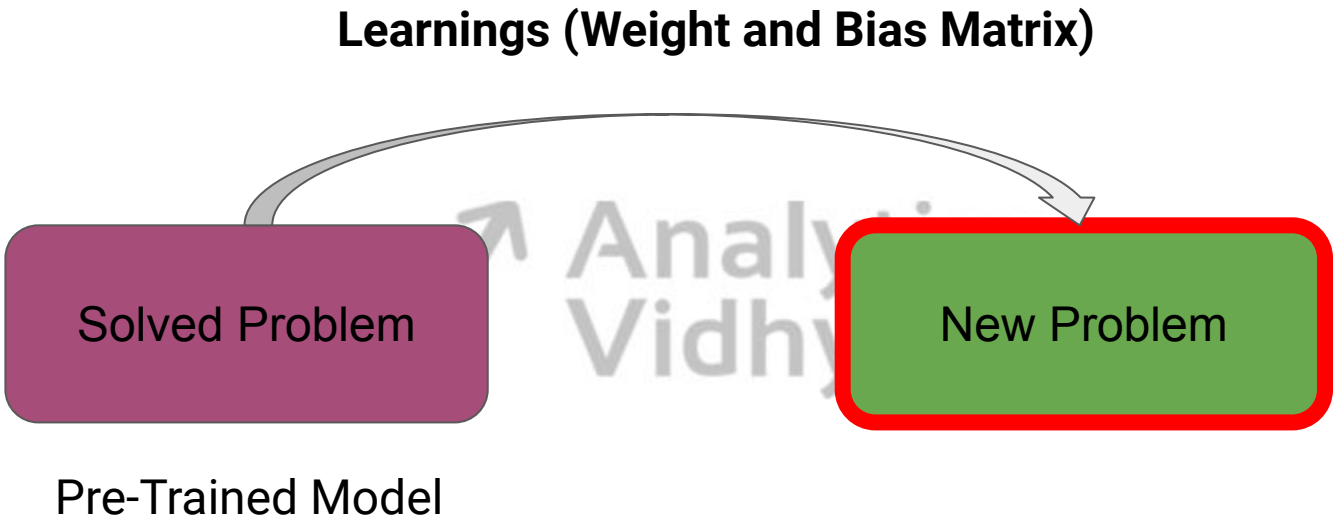
Transfer Learning



Transfer Learning



Transfer Learning



Selecting the right pre-trained model

BERT

VGG16 trained on ImageNet

ULMFiT

VGG16 trained on MNIST

Selecting the right pre-trained model

BERT

VGG16 trained on ImageNet

ULMFiT

VGG16 trained on MNIST

Different fine tuning techniques

1. Feature Extraction

Different fine tuning techniques

1. Feature Extraction

```
# creating a VGG16 model with imagenet pretrained weights , accepting input of shape (224,224,3)
# also remove the final layers from model(include_top= False)
base_model = VGG16(weights='imagenet', input_shape=(224, 224, 3), include_top=False)
```

Different fine tuning techniques

1. Feature Extraction

```
# creating a VGG16 model with imagenet pretrained weights , accepting input of shape (224,224,3)
# also remove the final layers from model(include_top= False)
base_model = VGG16(weights='imagenet', input_shape=(224, 224, 3), include_top=False)
```

```
# extract features using the pretrained VGG16 model
# for training set
base_model_pred = base_model.predict(X_train)
#for validation set
base_model_pred_valid = base_model.predict(X_valid)
```

Different fine tuning techniques

1. Feature Extraction

2. Using the Architecture of the pre-trained model

Different fine tuning techniques

1. Feature Extraction

2. Using the Architecture of the pre-trained model

```
# creating model with pre trained imagenet weights
base_model = VGG16(weights='imagenet')
```

Different fine tuning techniques

1. Feature Extraction

2. Using the Architecture of the pre-trained model

```
# creating model with pre trained imagenet weights
base_model = VGG16(weights='imagenet')
```

```
# creating our own model
x = Dense(100, activation='relu', name='fc1')(base_model.layers[-4].output)
y = Dense(2, activation='softmax', name='prediction')(x)
my_model = Model(input=base_model.input, output=y)
```

Different fine tuning techniques

1. Feature Extraction
2. Using the Architecture of the pre-trained model
3. Train some layers while freeze others

Different fine tuning techniques

1. Feature Extraction
2. Using the Architecture of the pre-trained model
3. Train some layers while freeze others


```
# creating model with pre trained imagenet weights
base_model = VGG16(weights='imagenet')
```

Different fine tuning techniques

1. Feature Extraction
2. Using the Architecture of the pre-trained model
3. Train some layers while freeze others


```
# creating model with pre trained imagenet weights
base_model = VGG16(weights='imagenet')
```

```
# creating our own model
x = Dense(100, activation='relu', name='fc1')(base_model.layers[-4].output)
y = Dense(2, activation='softmax', name='prediction')(x)
my_model = Model(input=base_model.input, output=y)
```

Different fine tuning techniques

1. Feature Extraction
2. Using the Architecture of the pre-trained model
3. Train some layers while freeze others


```
# creating model with pre trained imagenet weights
base_model = VGG16(weights='imagenet')
```

```
# creating our own model
x = Dense(100, activation='relu', name='fc1')(base_model.layers[-4].output)
y = Dense(2, activation='softmax', name='prediction')(x)
my_model = Model(input=base_model.input, output=y)
```

```
# to set the first 15 layers to non-trainable (weights will not be updated)
for layer in my_model.layers[:15]:
    layer.trainable = False
```

Thank You