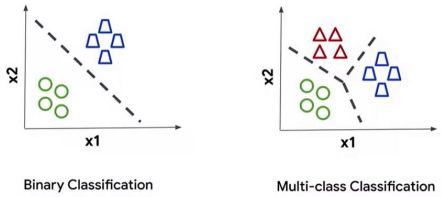
Image Classification

Image Classification

- Multi-class Classification
  - Binary Classification (Subset of the problem)
- Multi-label Classification

Multi-label Classification

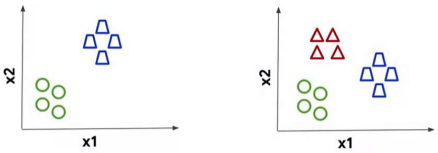
Binary vs. Multi Class Classification



Binary Classification

Multi-class Classification

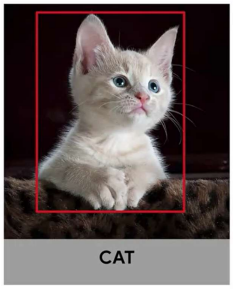
Binary vs. Multi Class Classification



Binary Classification

Multi-class Classification

Object Localization



Object Detection

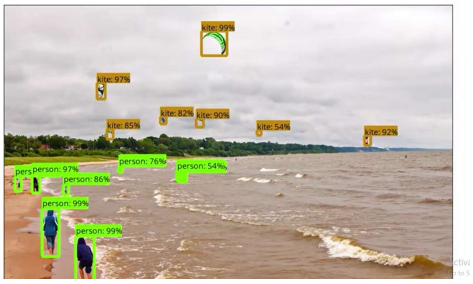


Image Segmentation

Object Detection

- For each object:
  - confidence scores
  - bounding boxes.
- Popular algorithms
  - R-CNN
  - Faster-RCNN
  - YOLO
  - SSD

Semantic vs. Instance Segmentation

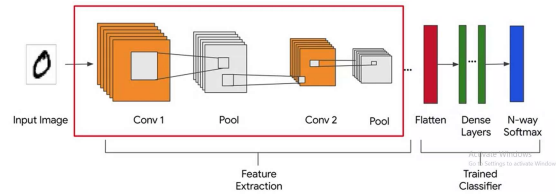
Semantic Segmentation

Instance Segmentation

Semantic Segmentation

- Same class → one segment.
- Each pixel is associated with one class.
  - All person(s) in an image are treated as one segment
- Popular models are Fully Convolutional Neural Networks, U-Net, DeepLab

Convolutional Neural Networks Architecture

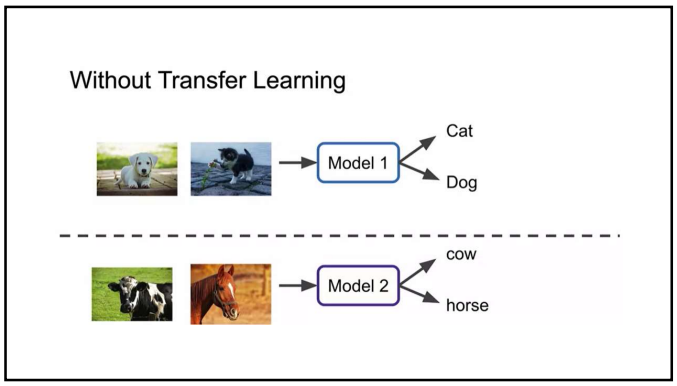


Instance Segmentation

- Multiple "instances" of same class are separate segments.

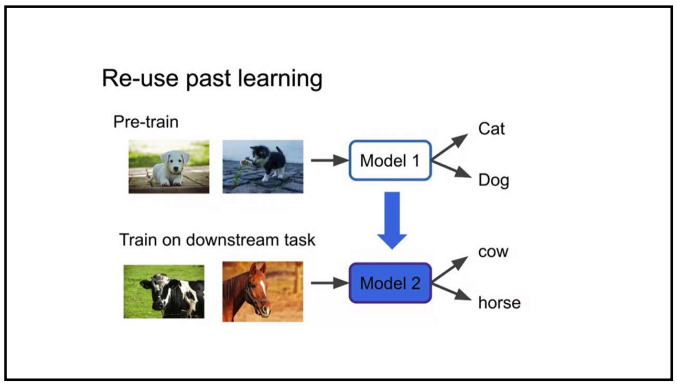
- Popular algorithms are Mask R-CNN.

Without Transfer Learning

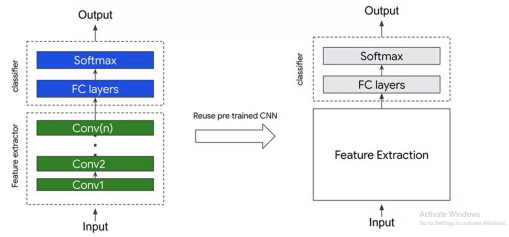


What is Transfer Learning?

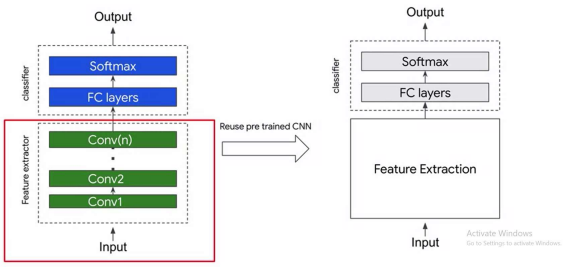
- Pre trained model reused for solving downstream task.
- Reuse weights and layers.
- Example: pre-trained MobileNetV2 → cats vs dog classifier.



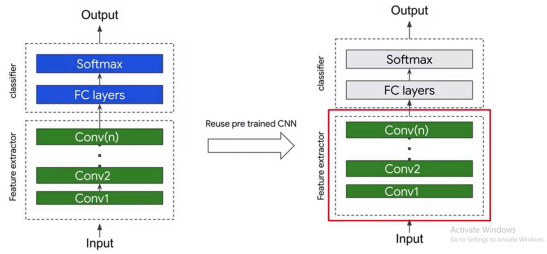
What is Transfer Learning?



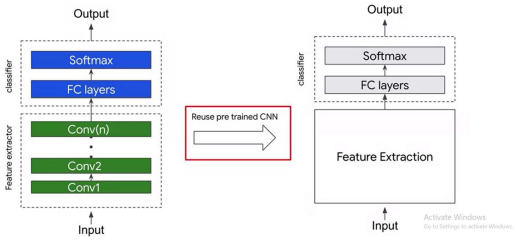
What is Transfer Learning?



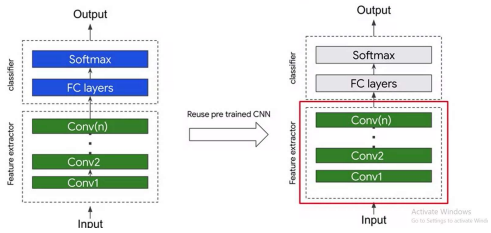
What is Transfer Learning?



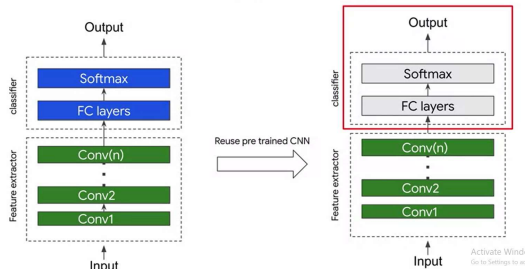
What is Transfer Learning?



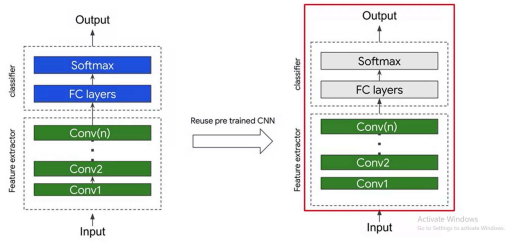
Freeze the Weights



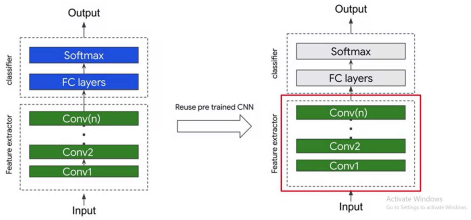
Freeze the Weights



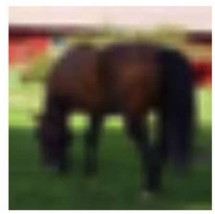
Train with learned values as default



Train with learned values as default



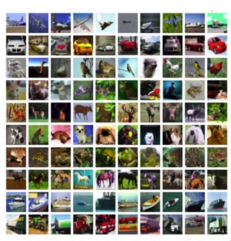
Transfer Learning with ResNet50



CIFAR - 10 Dataset

10

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



https://www.cs.toronto.edu/~kriz/cifar.html

~6,000

...

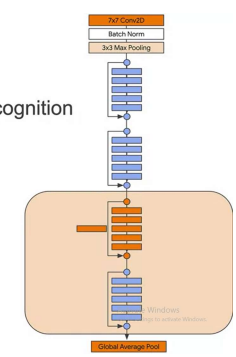
~6,000

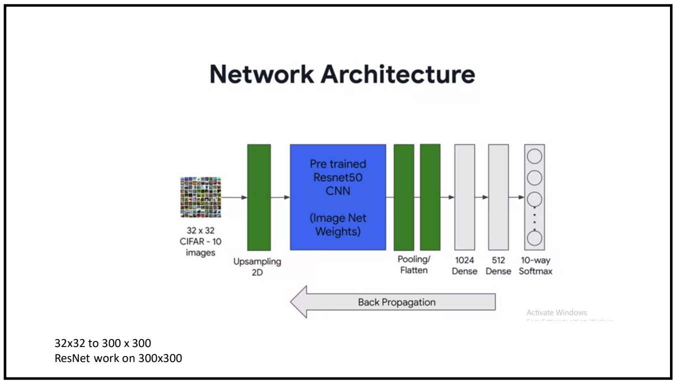
= 60,000

ResNet 50

Deep Residual Learning for Image Recognition

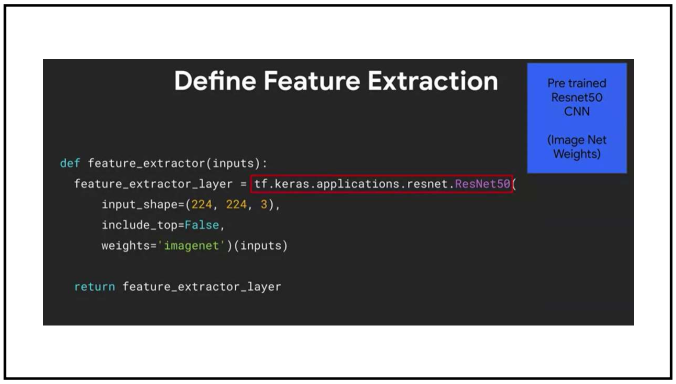
https://arxiv.org/abs/1512.03385





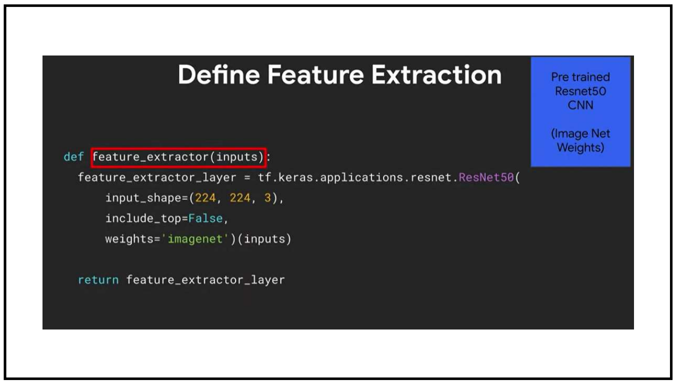
Network Architecture

32x32 to 300 x 300
ResNet work on 300x300



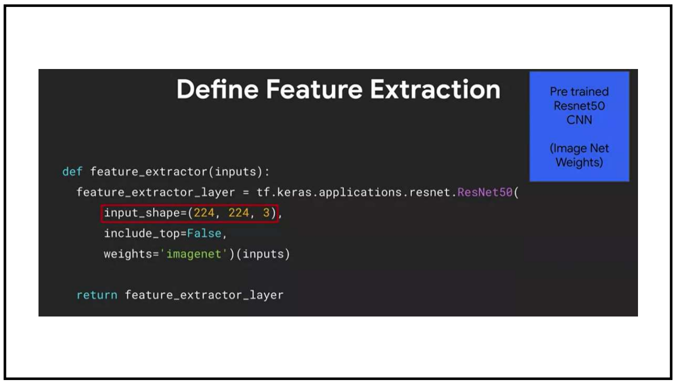
Define Feature Extraction

```
def feature_extractor(inputs):
  feature_extractor_layer = tf.keras.applications.resnet.ResNet50(
      input_shape=(224, 224, 3),
      include_top=False,
      weights='imagenet')(inputs)

  return feature_extractor_layer
```



Define Feature Extraction

```
def feature_extractor(inputs):
  feature_extractor_layer = tf.keras.applications.resnet.ResNet50(
      input_shape=(224, 224, 3),
      include_top=False,
      weights='imagenet')(inputs)

  return feature_extractor_layer
```



Define Feature Extraction

```
def feature_extractor(inputs):
  feature_extractor_layer = tf.keras.applications.resnet.ResNet50(
      input_shape=(224, 224, 3),
      include_top=False,
      weights='imagenet')(inputs)

  return feature_extractor_layer
```

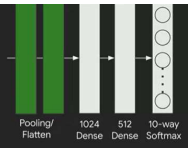
Define Feature Extraction

Pre trained Resnet50 CNN

(Image Net Weights)

```
def feature_extractor(inputs):
  feature_extractor_layer = tf.keras.applications.resnet.ResNet50(
      input_shape=(224, 224, 3),
      include_top=False,
      weights='imagenet')(inputs)

  return feature_extractor_layer
```

Define Classifier




```
def classifier(inputs):
    x = tf.keras.layers.GlobalAveragePooling2D()(inputs)
    x = tf.keras.layers.Flatten()(x)
    x = tf.keras.layers.Dense(1024, activation="relu")(x)
    x = tf.keras.layers.Dense(512, activation="relu")(x)
    x = tf.keras.layers.Dense(10, activation="softmax", name="classification")(x)
    return x
```

Define Feature Extraction

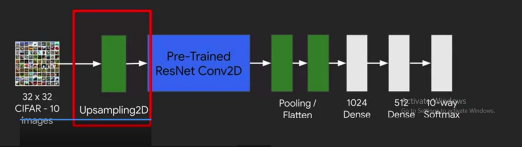
Pre trained Resnet50 CNN

(Image Net Weights)

```
def feature_extractor(inputs):
  feature_extractor_layer = tf.keras.applications.resnet.ResNet50(
      input_shape=(224, 224, 3),
      include_top=False,
      weights='imagenet')(inputs)

  return feature_extractor_layer
```

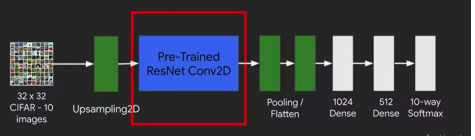
Define Inputs and Outputs

```
def final_model(inputs):
    resize = tf.keras.layers.UpSampling2D(size=(7,7))(inputs)
```



Define Inputs and Outputs

```
def final_model(inputs):
    resize = tf.keras.layers.UpSampling2D(size=(7,7))(inputs)

    resnet_feature_extractor = feature_extractor(resize)
```



Define Inputs and Outputs

```
def final_model(inputs):
    resize = tf.keras.layers.UpSampling2D(size=(7,7))(inputs)

    resnet_feature_extractor = feature_extractor(resize)
    classification_output = classifier(resnet_feature_extractor)

    return classification_output
```