Track 7

Inteligencia Artificial Aplicada

8a. Preventing Overfitting

Prof. Marcelo José Rovai
rovai@unifei.edu.br

UNIFEI - Universidade Federal de Itajubá, Brazil

Preventing Overfitting

+Data

+Data

+Data

+Data

Preventing Overfitting

+Data

+Data

+Data

+Data

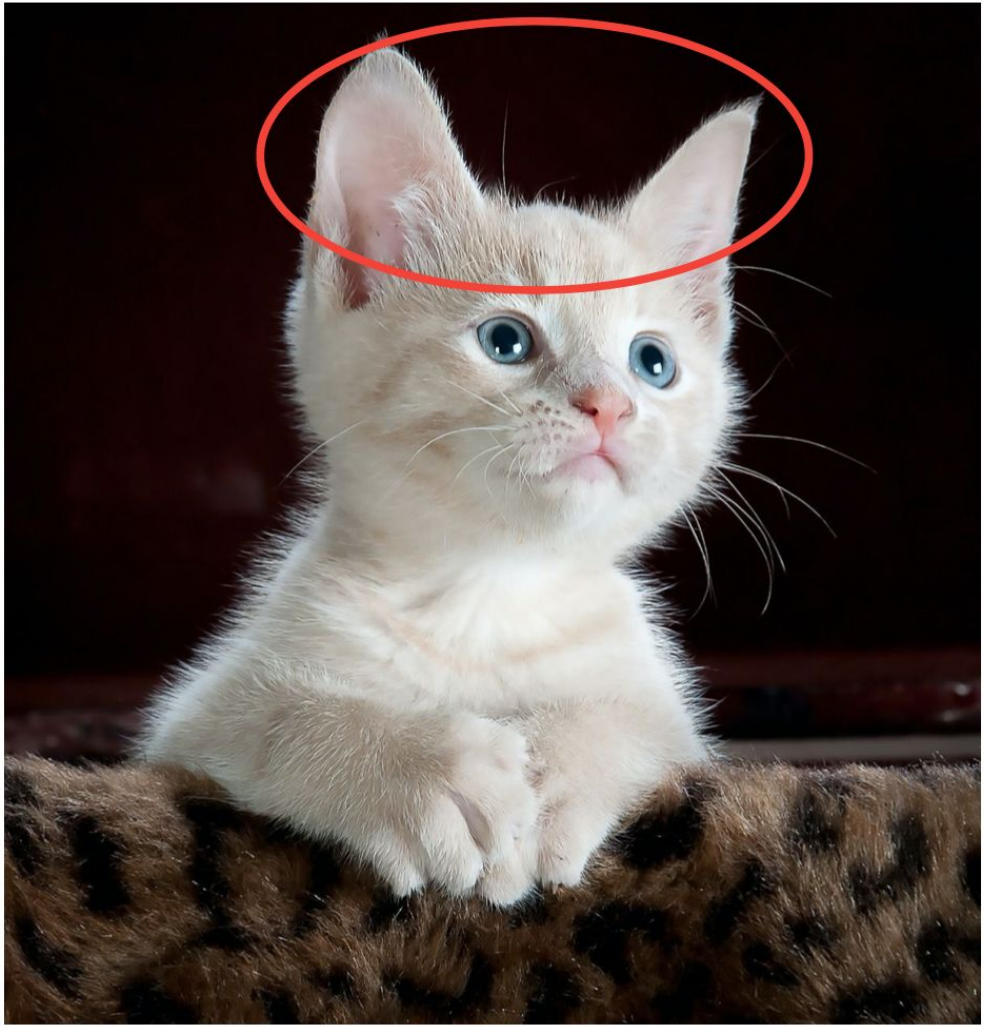
What can we do if we don't have enough data?

- Data Augmentation (artificial)
- Transfer Learning
- Early Stopping
- Dropout Regularization

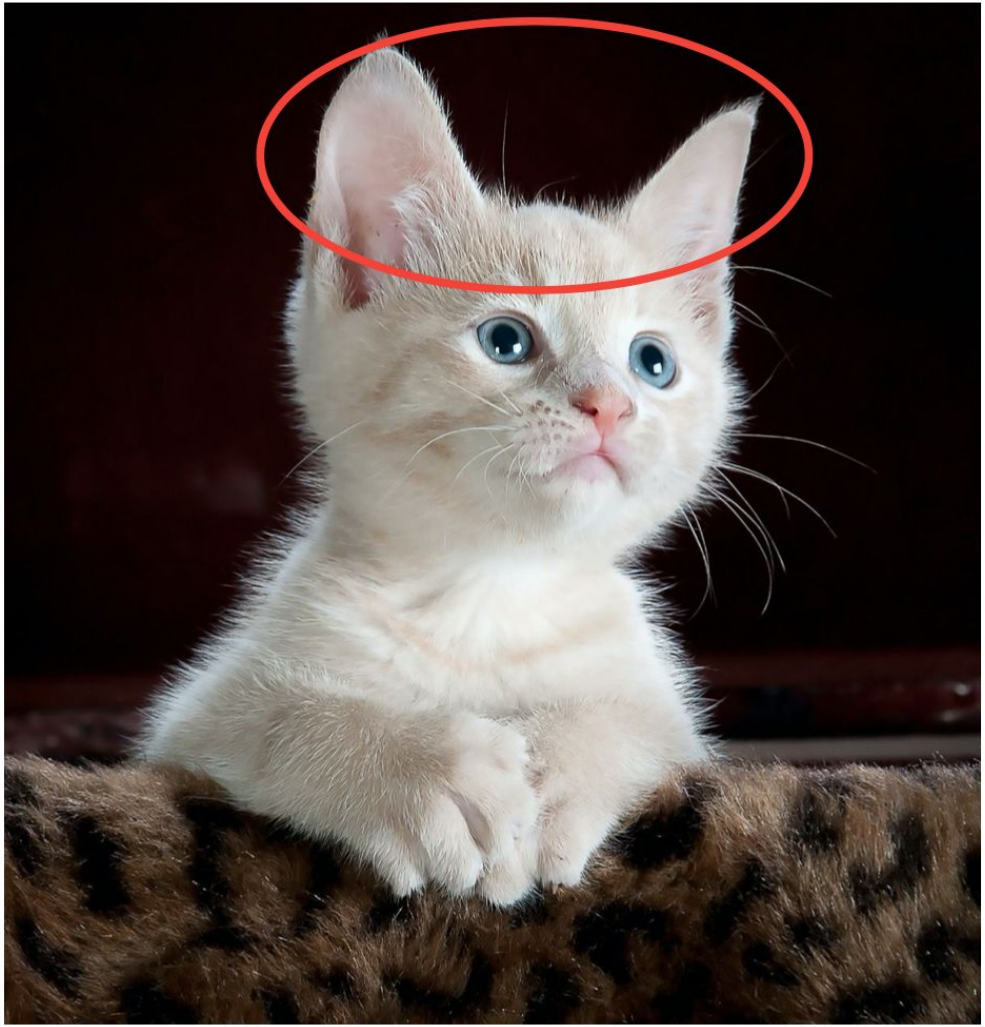
Preventing Overfitting

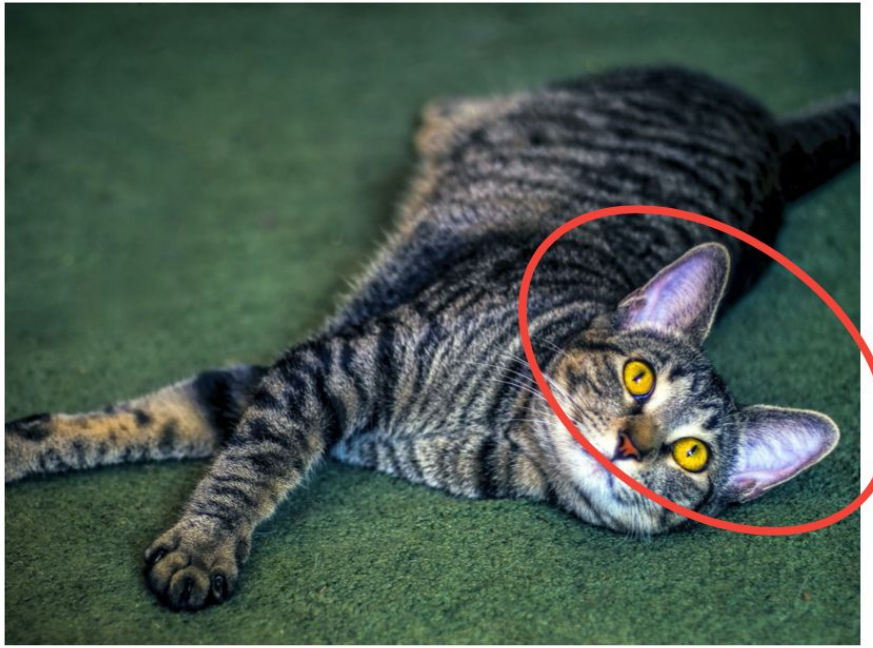
More Data, Data Augmentation (artificial)

Overfitting generally occurs when there are a small number of training examples. Data augmentation takes the approach of generating additional training data from your existing examples by augmenting them using random transformations that yield believable-looking images. This helps expose the model to more aspects of the data and generalize better.



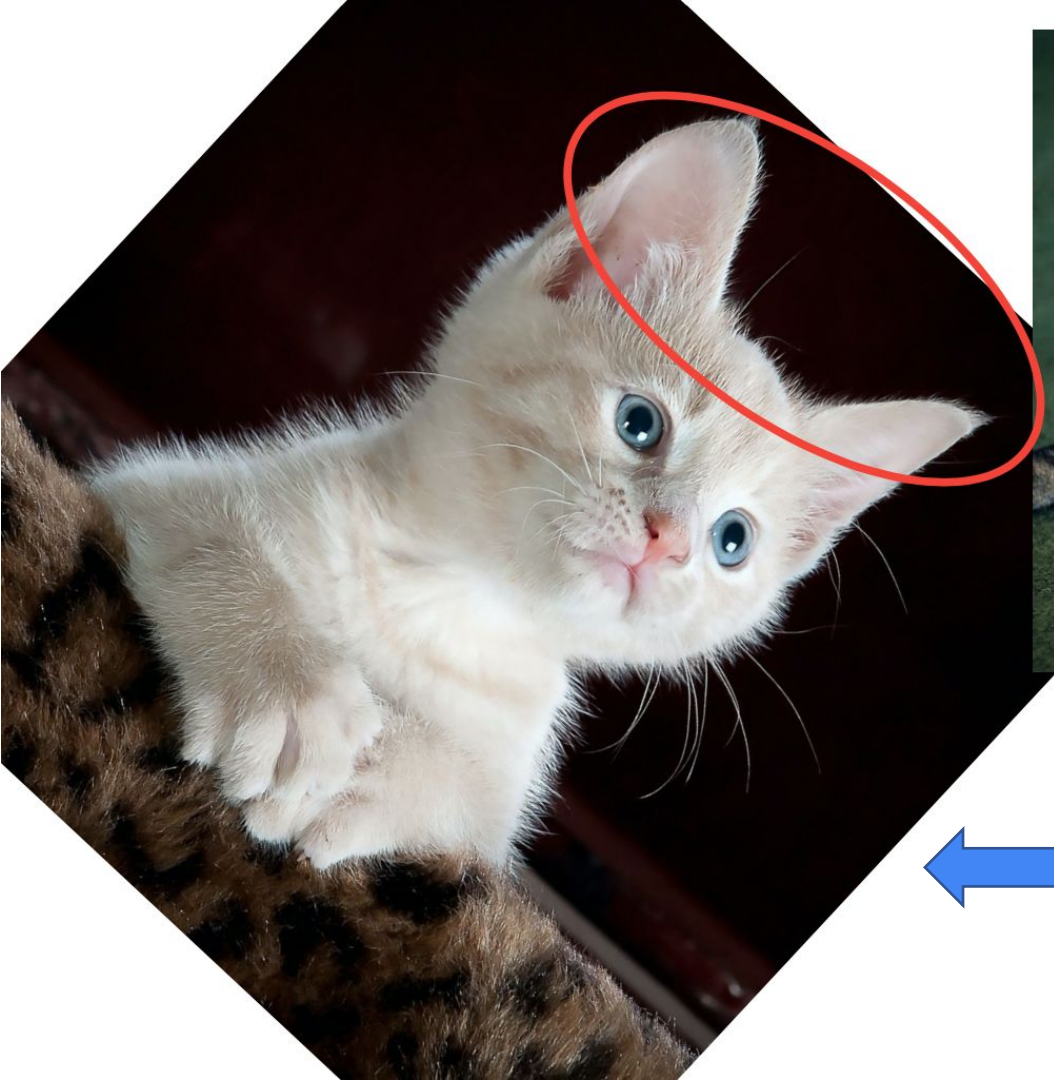
Training

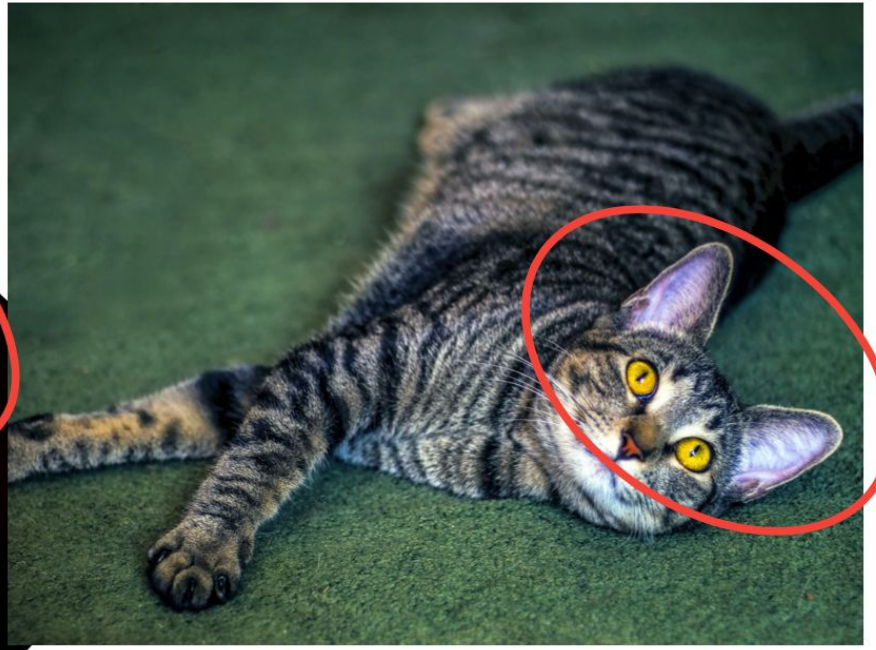




Inference

Training



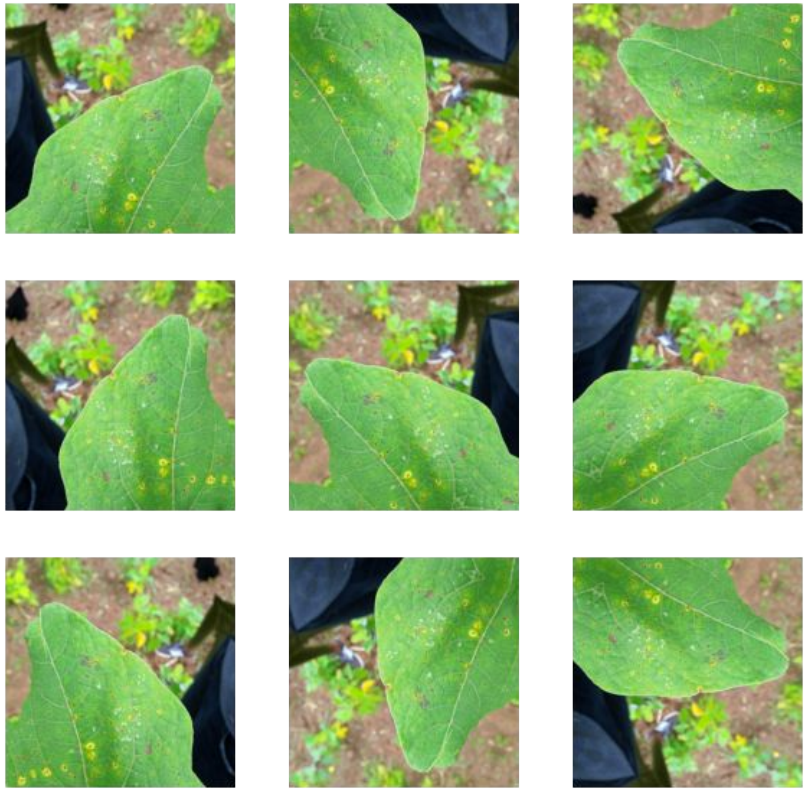


Training with
Data Augmentation

Using Keras preprocessing layers

```
data_augmentation = tf.keras.Sequential([
  layers.RandomFlip("horizontal_and_vertical"),
  layers.RandomRotation(0.2),
])
```

```
plt.figure(figsize=(10, 10))
for i in range(9):
  augmented_image = data_augmentation(image)
  ax = plt.subplot(3, 3, i + 1)
  plt.imshow(augmented_image[0])
  plt.axis("off")
```

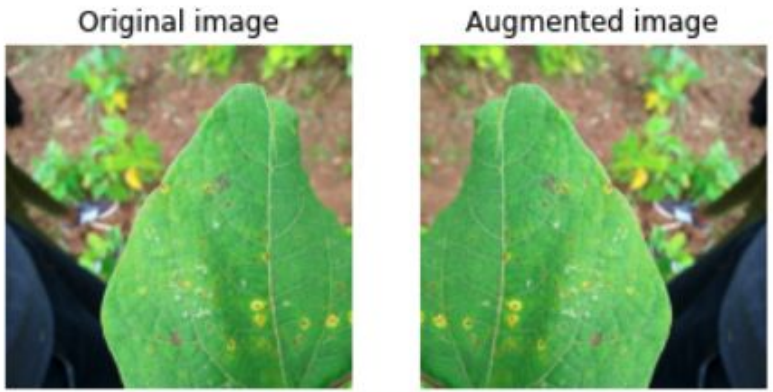


There are a variety of preprocessing layers you can use for data augmentation including:

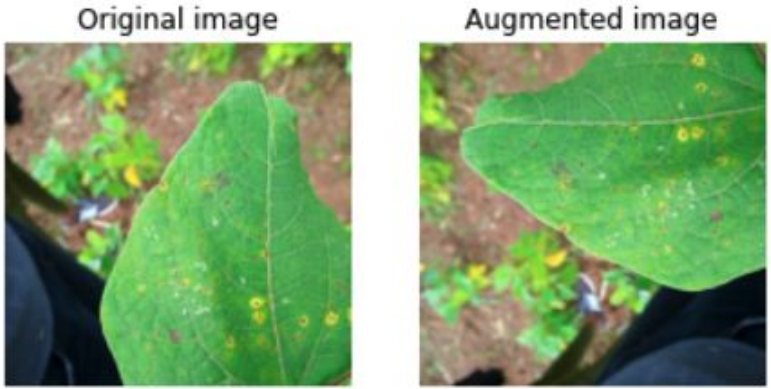
- tf.keras.layers.RandomContrast,
- tf.keras.layers.RandomCrop,
- tf.keras.layers.RandomZoom,
- and others.

Using tf.image

```
1 flipped = tf.image.flip_left_right(image)
2 visualize(image, flipped)
```

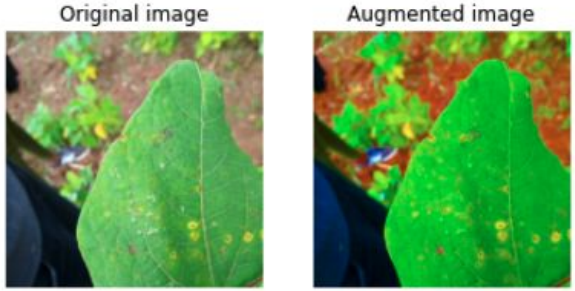



```
1 rotated = tf.image.rot90(image)
2 visualize(image, rotated)
```

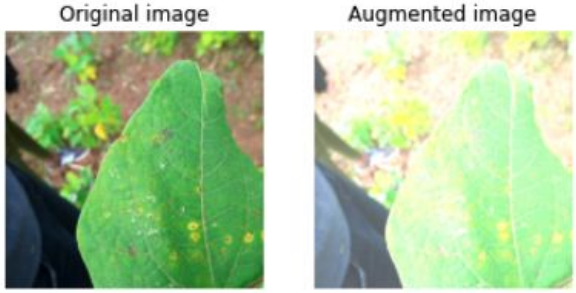


Using tf.image

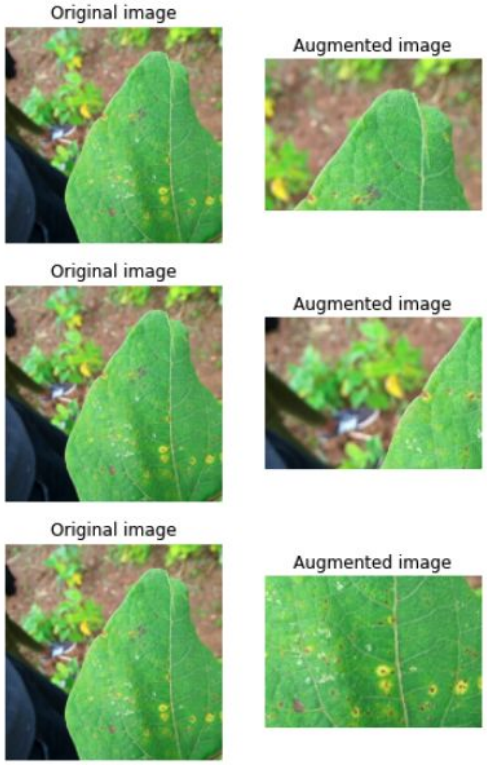
```
saturated = tf.image.adjust_saturation(image, 3)
visualize(image, saturated)
```




```
bright = tf.image.adjust_brightness(image, 0.4)
visualize(image, bright)
```



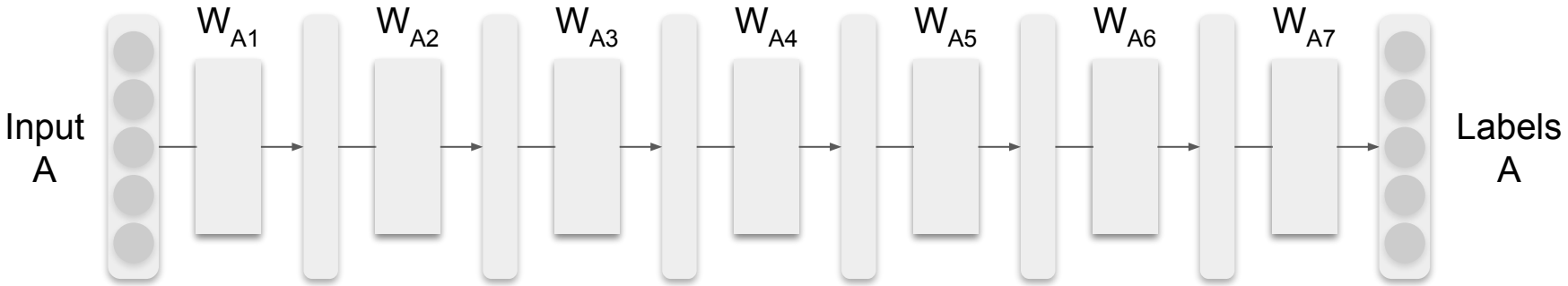

```
for i in range(3):
  seed = (i, 0)  # tuple of size (2,)
  stateless_random_crop = tf.image.stateless_random_crop(
      image, size=[210, 300, 3], seed=seed)
  visualize(image, stateless_random_crop)
```



Preventing Overfitting

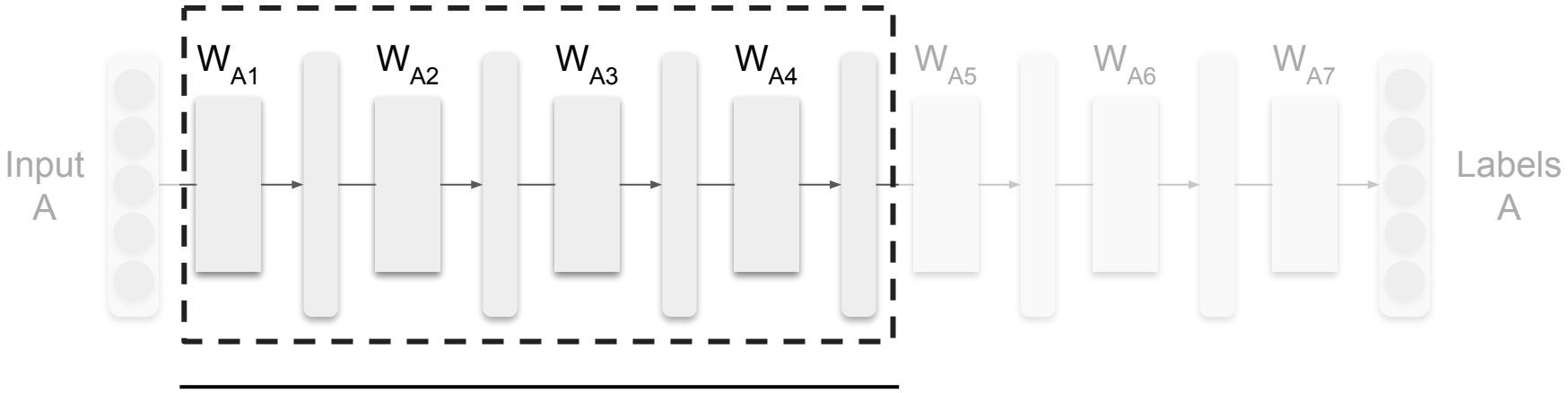
Transfer Learning

End Result of Training



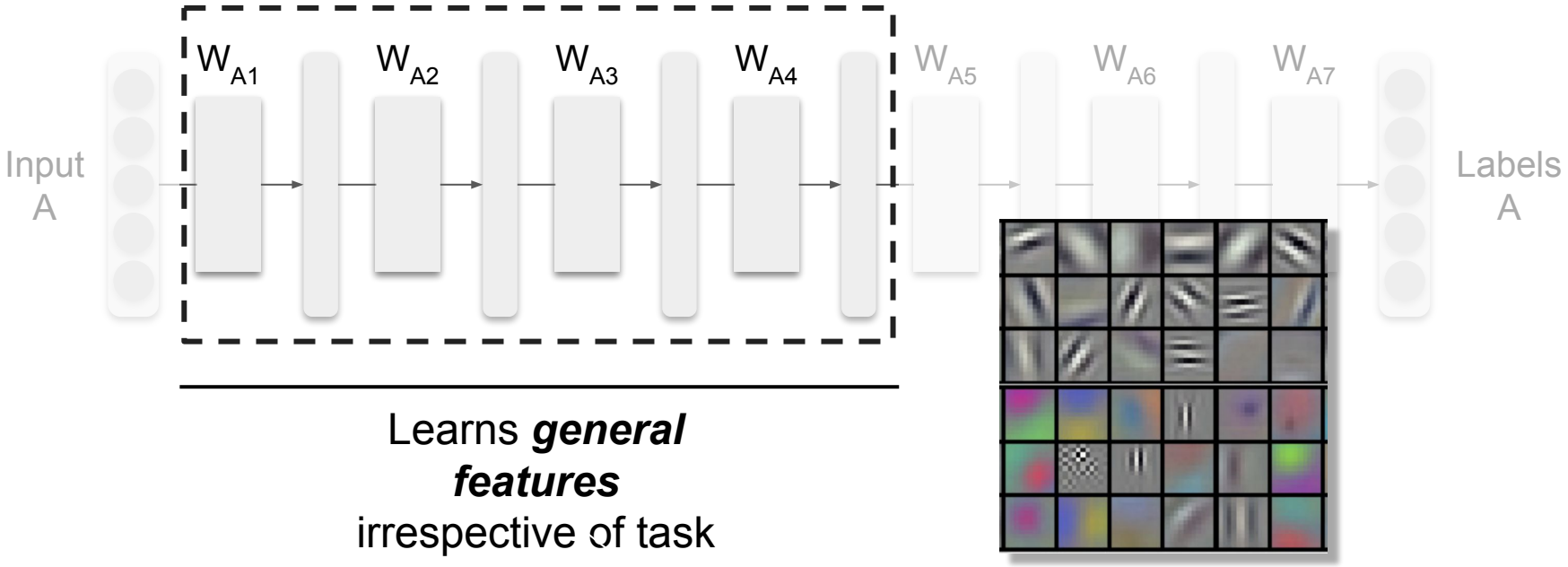
The end result of the training is to learn the weights of the neural network model.

End Result of Training

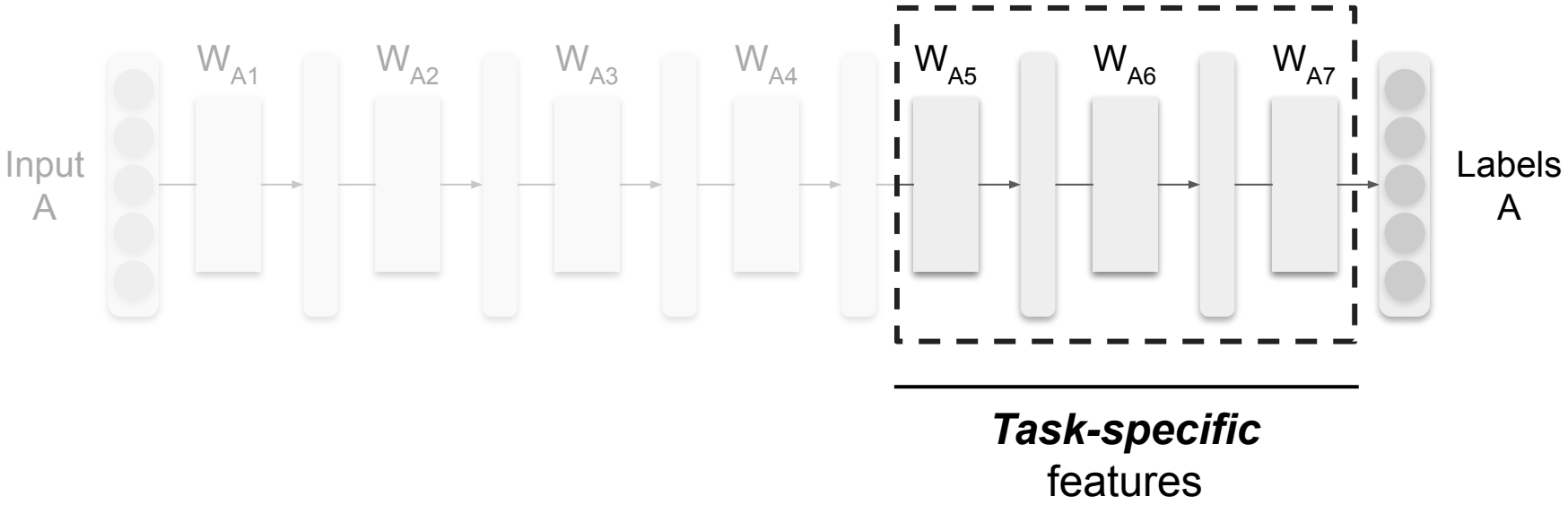


Learns ***general features*** irrespective of task

End Result of Training



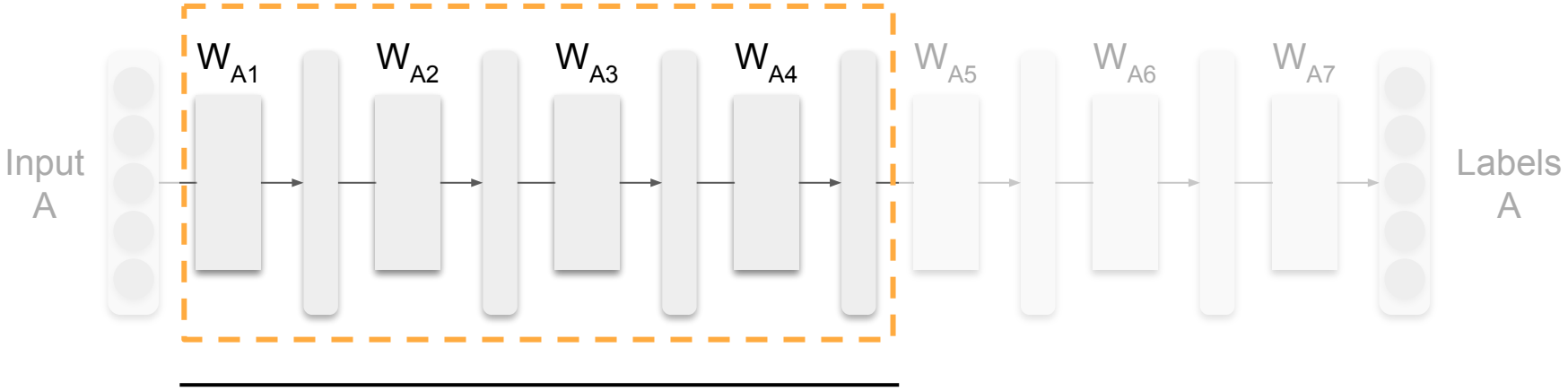
End Result of Training



Source: Google

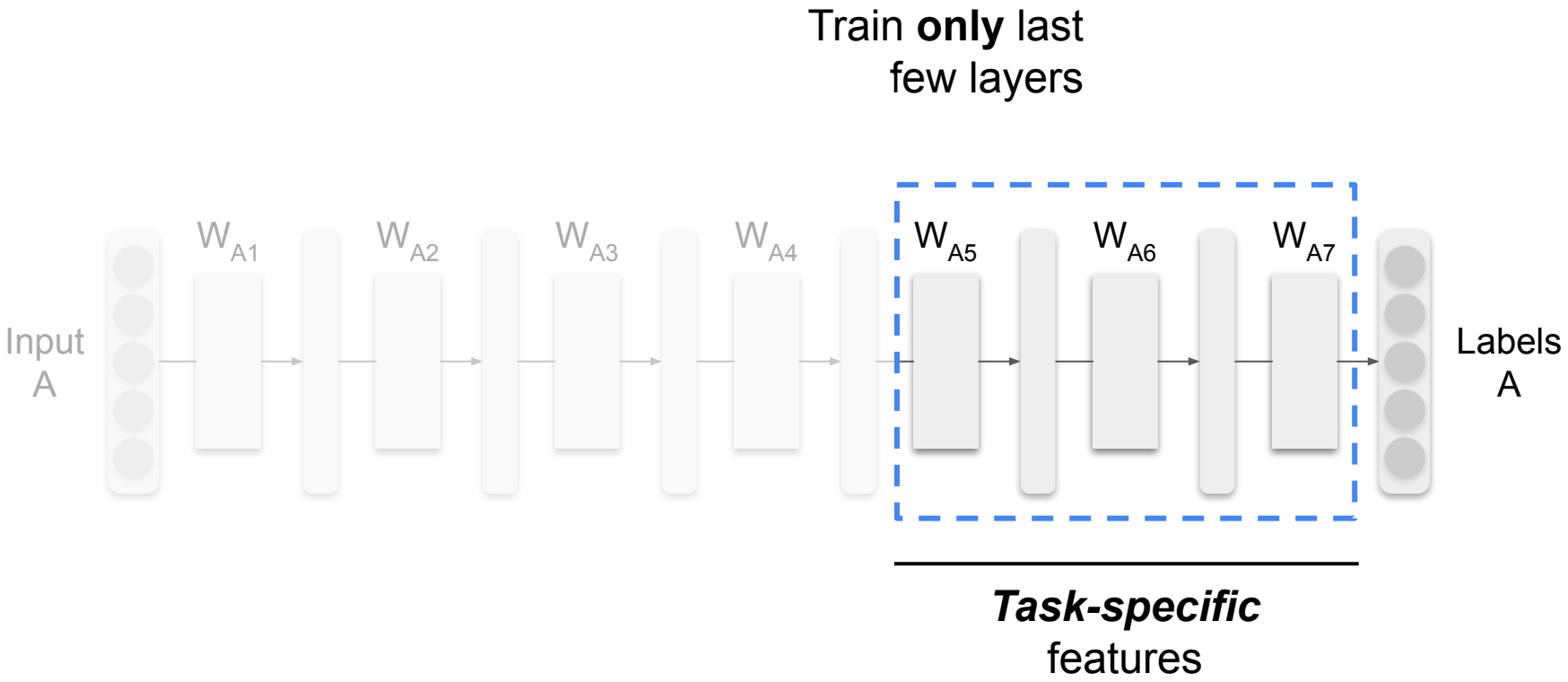
Transfer Learning

Reuse (freeze general feature extraction)



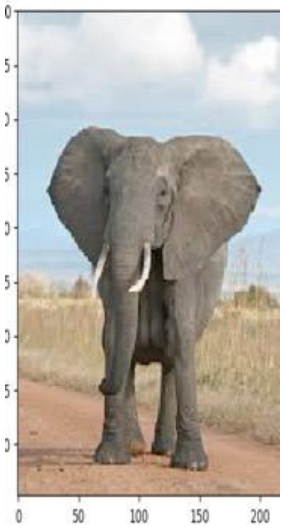
Learns ***general features*** irrespective of task

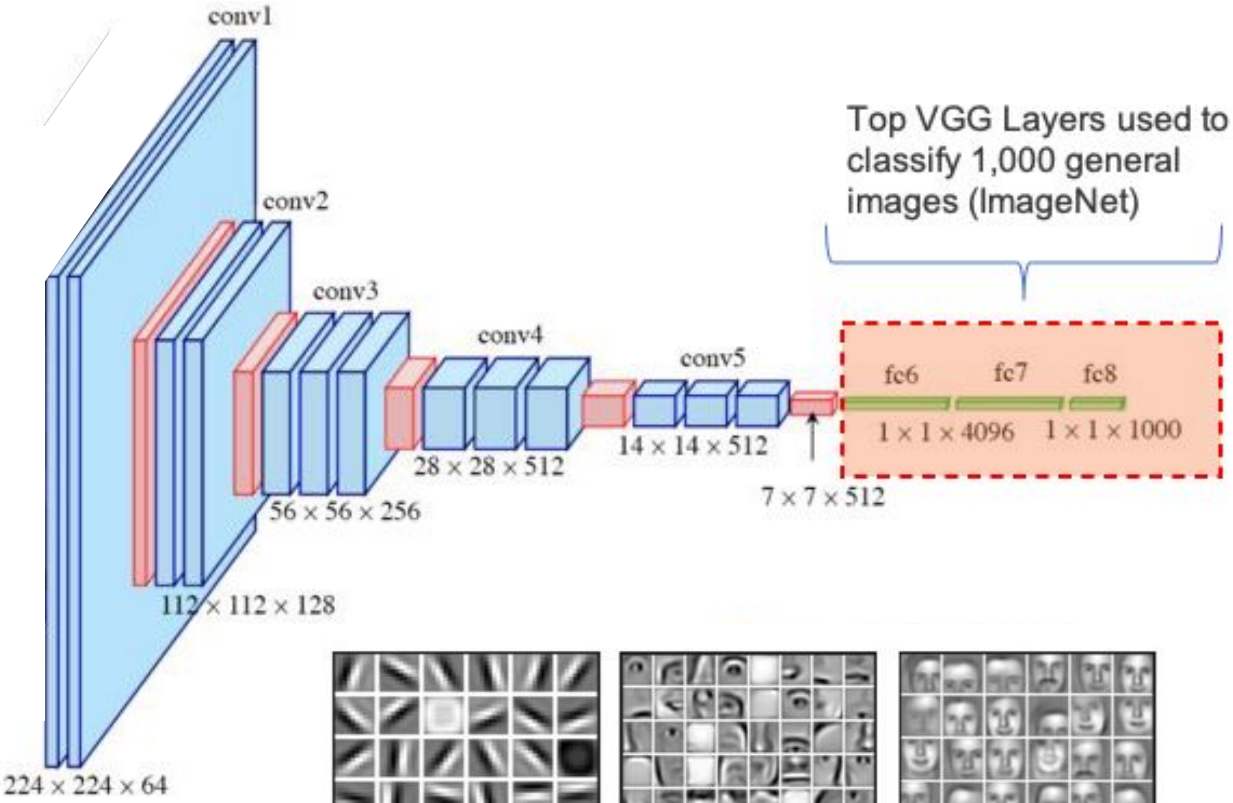
Transfer Learning



covidXray
Detecting Covid-19 in Chest X-Ray images

VGG-16 Convolutional Neural Network Model

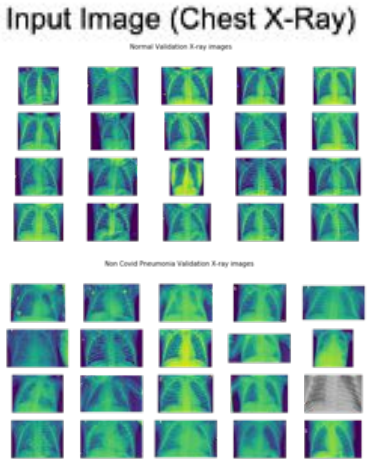


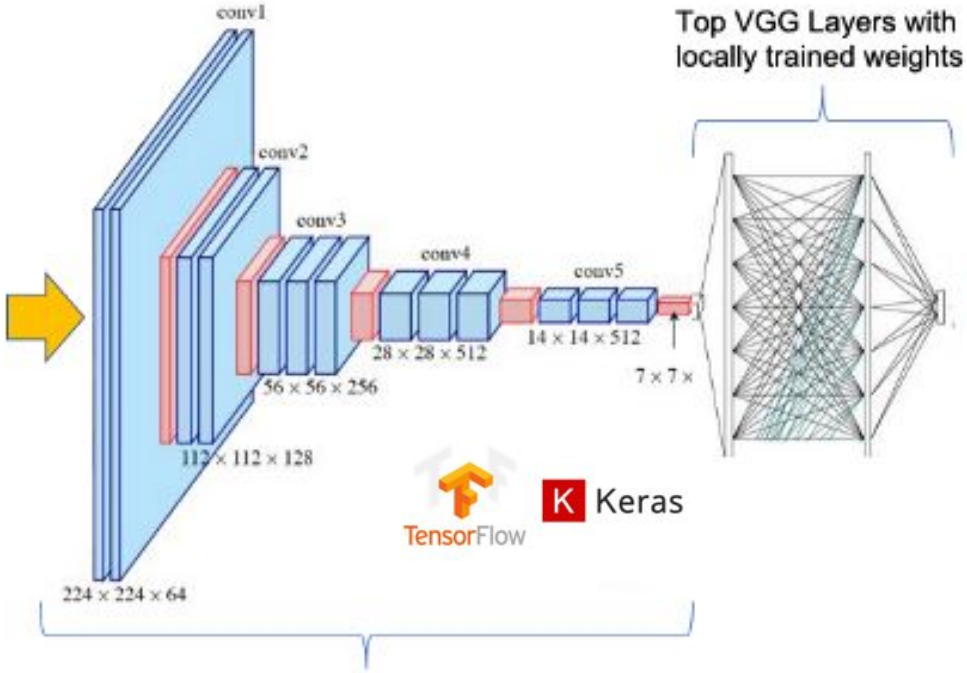


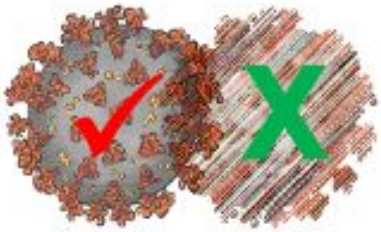
Elephant
93.0%

First Layer Representation | Second Layer Representation | Third Layer Representation

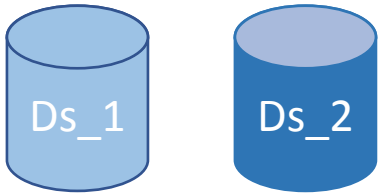
Training the model (Transfer Learning)



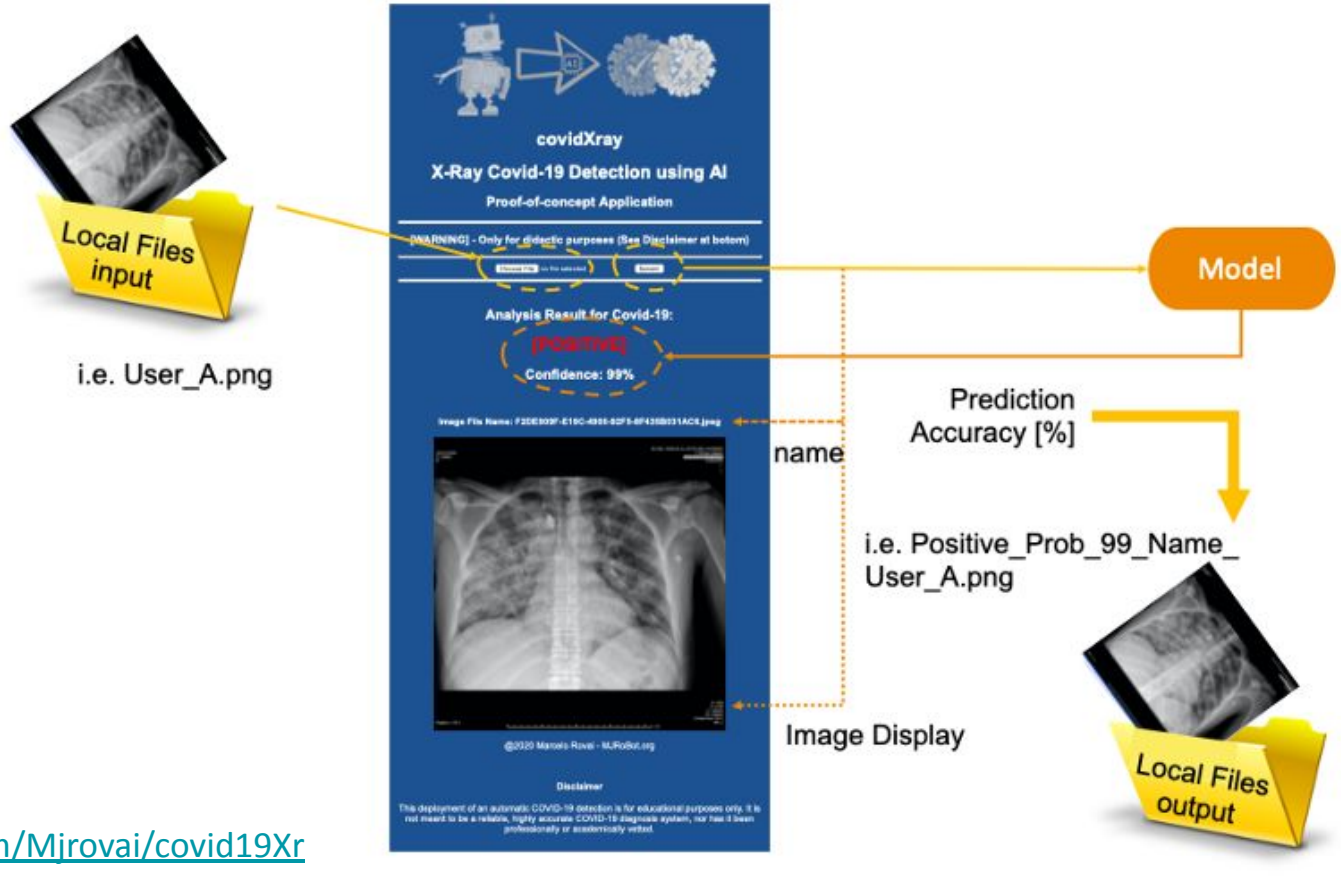




Classify:
Positive/Negative
for Covid-19



Inference



https://github.com/Mjrovai/covid19Xray

Preventing Overfitting

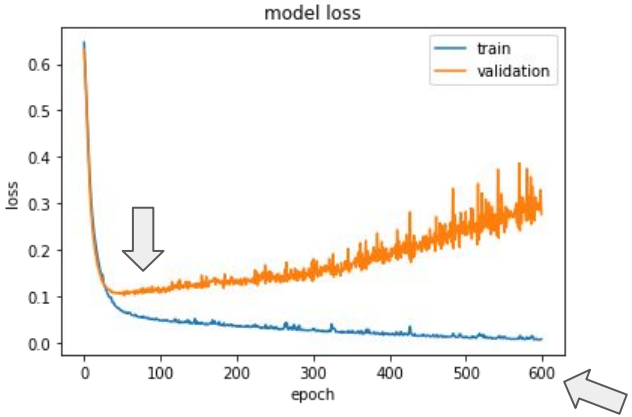
Early Stopping & Dropout Regularization

Early Stopping

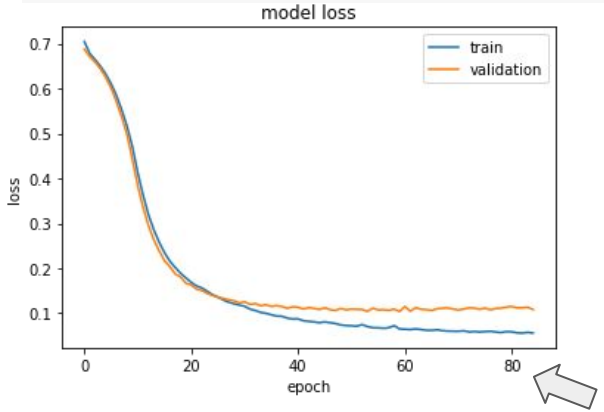
```
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss',
                           mode='min',
                           verbose=1,
                           patience=25)
```

```
history = model.fit(X_train,
                    y_train,
                    epochs=600,
                    validation_data=(X_test, y_test),
                    verbose=1
                    )
```

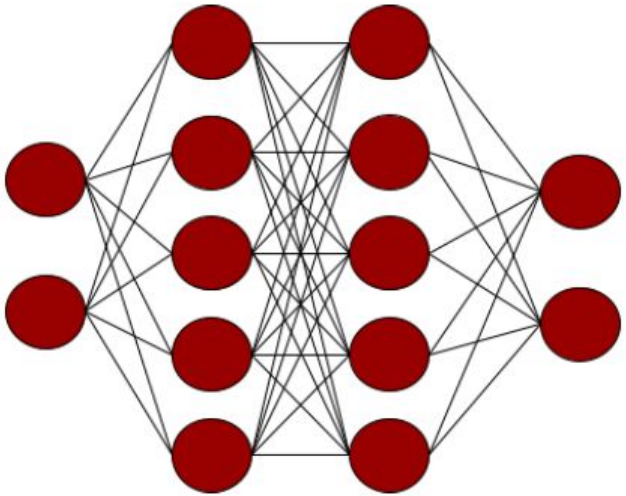



```
history = model.fit(x=X_train,
          y=y_train,
          epochs=600,
          validation_data=(X_test, y_test),
          verbose=1,
          callbacks=[early_stop]
          )
```



Dropout Regularization

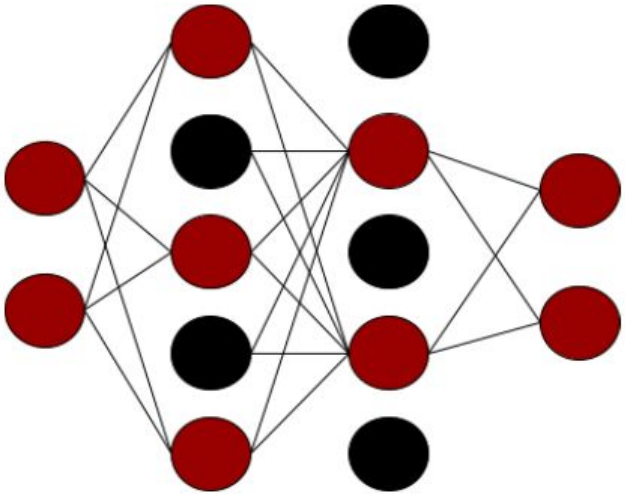
```
model = tf.keras.models.Sequential([
                        tf.keras.layers.Flatten(input_shape=(28,28)),
                        tf.keras.layers.Dense(256, activation=tf.nn.relu),
                        tf.keras.layers.Dense(128, activation=tf.nn.relu),
                        tf.keras.layers.Dense(64, activation=tf.nn.relu),
                        tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```



Fashion MNIST Dataset

- 20 Epochs
- 94.0% Accuracy on Train Data
- 88.5% Accuracy on Validation Data

Dropout Regularization



```
model = tf.keras.models.Sequential([
                    tf.keras.layers.Flatten(input_shape=(28,28)),
                    tf.keras.layers.Dense(256, activation=tf.nn.relu),
                    tf.keras.layers.Dropout(0.2),
                    tf.keras.layers.Dense(128, activation=tf.nn.relu),
                    tf.keras.layers.Dropout(0.2),
                    tf.keras.layers.Dense(64, activation=tf.nn.relu),
                    tf.keras.layers.Dropout(0.2),
                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

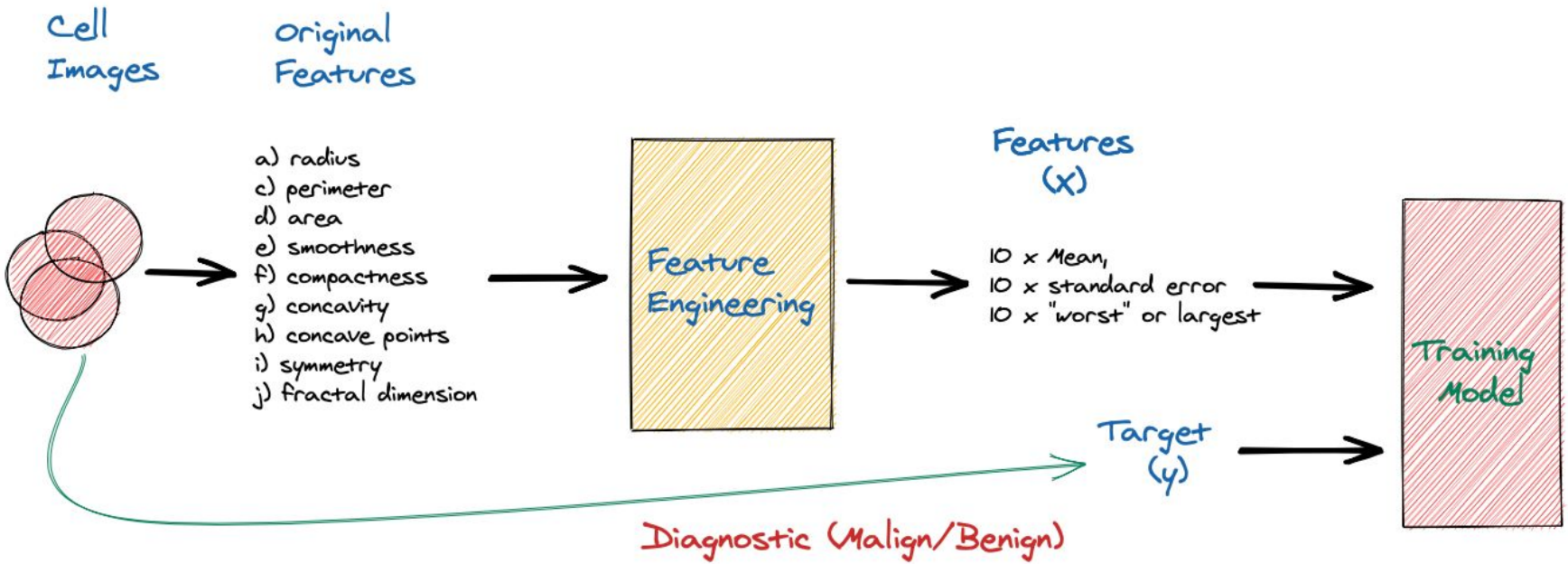
Fashion MNIST Dataset

- 20 Epochs
- 89.5% Accuracy on Train Data
- 88.3% Accuracy on Validation Data

Removing a random number of neurons and connections (in this example, 20%), reduces the chances of the neurons becoming overspecialized and the model will generalize better, reducing the overfit.

Wisconsin Diagnostic Breast Cancer (WDBC)
Code Time!

Breast_Cancer_Classification.ipynb



UCI ML Breast Cancer Wisconsin (Diagnostic) datasets. https://goo.gl/U2Uwz2

Thanks

UNIFEI