Setup

Import TensorFlow and other necessary libraries:

```
import matplotlib.pyplot as plt
import numpy as np
import PIL
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

import pandas as pd
```

Import Data

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pathlib
data_dir = pathlib.Path('/content/drive/MyDrive/is-that-santa')
```

There are 1230 total images:

```
image_count = len(list(data_dir.glob('*/*.jpg')))
print(image_count)

santa_count = len(list(data_dir.glob('santa/*')))
print(santa_count)

notSanta_count = len(list(data_dir.glob('not-a-santa/*')))
print(notSanta_count)
```

```
1230
615
615
```

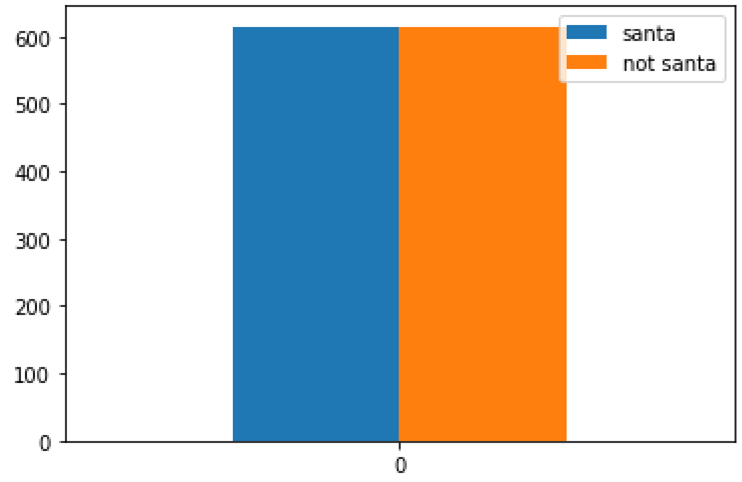
Target Distribution

```
# Distribution
df = pd.DataFrame([(santa_count, notSanta_count)], columns=('santa', 'not santa'))

df.plot.bar(rot=0)

# It's even so it doesn't really matter
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcede1a5730>
```



Here are some santas:

```
santa = list(data_dir.glob('santa/*'))
PIL.Image.open(str(santa[2]))
```


```
PIL.Image.open(str(santa[1]))
```

And some not santas:

```
notSanta = list(data_dir.glob('not-a-santa/*'))
PIL.Image.open(str(notSanta[0]))
```

```
PIL.Image.open(str(notSanta[1]))
```

Load data using a Keras utility

Load these images off disk using `tf.keras.utils.image_dataset_from_directory` utility.

[ ] ↳ 7 cells hidden

Visualize the data

Here are the first nine images from the training dataset:

[ ] ↳ 1 cell hidden

Configure the dataset for performance

Using `Dataset.cache` and `Dataset.prefetch`

[ ] ↳ 1 cell hidden

Standardize the data

[ ] ↳ 4 cells hidden

A Basic Keras Sequential Model

Create the model

The Keras Sequential model consists of three convolution blocks (`tf.keras.layers.Conv2D`) with a max pooling layer (`tf.keras.layers.MaxPooling2D`) in each of them. There's a fully-connected layer (`tf.keras.layers.Dense`) with 128 units on top of it that is activated by a ReLU activation function (`'relu'`).

[ ] ↳ 8 cells hidden

Visualize training results

[ ] ↳ 3 cells hidden

Data augmentation to handle Overfitting (Adjusted CNN)

[ ] ↳ 4 cells hidden

Dropout

Dropout Regularization

Create a new neural network with `tf.keras.layers.Dropout`

[ ] ↳ 1 cell hidden

Compile and train the model

[ ] ↳ 3 cells hidden

Visualize training results

[ ] ↳ 1 cell hidden

Model Transfer

Data Pre-processing

[ ] ↳ 1 cell hidden

Rescale pixels

[ ] ↳ 1 cell hidden

Create Base Model from MobileNet V2

[ ] ↳ 18 cells hidden

Visualize the Training Results

[ ] ↳ 2 cells hidden

Fine tuning

[ ] ↳ 3 cells hidden

Recompile Model

[ ] ↳ 3 cells hidden

Retrain Model

[ ] ↳ 1 cell hidden

Visualize the Results

[ ] ↳ 2 cells hidden

Conclusions

I had 4 main approaches (4 models) that I worked with in this notebook. They were all sequential models and there was a regular CNN, a CNN with data augmentation, a model transfer, and model transfer with tuning.

The first CNN did amazingly well with the train set and managed to perfectly identify santa/not-santa. Even the validation accuracy of .93 was really impressive, but that tells me that this data was probably really easy to seperate.

The second CNN with data augmentation did worse when I ran it but the accuracy and loss graphs were much closer together than the first model. This could hopefully imply it could better generalize to other data it has yet to see.

The first model transfer did even poorer that the second CNN with its validation accuracy just trailing behind the second CNN. However, this was without any tuning.

The tuned model transfer did fantastic, even better than the first CNN when considering the validation accuracy of .96.

I assume these models did so well because the test data mostly consists of people or things that look like people. Given data that isn't similar to what I was using, I think the models would do significantly worse.