Crowd Counting

- Anwesha Tomar, Jingya Gao, Wenyu Zeng

Introduction:

- Source Data:

Images collected from a webcam in a mall

- Problem statement:

Object detection problem that counts the number of people in each image.

- Project Design:

TensorFlow framework

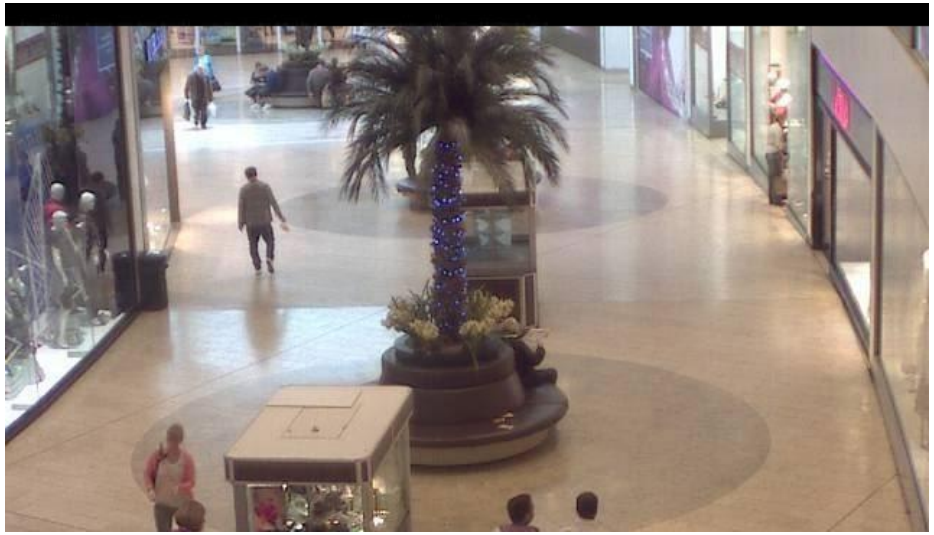
Comparing CNN model with a fine tuned ResNet 50 model

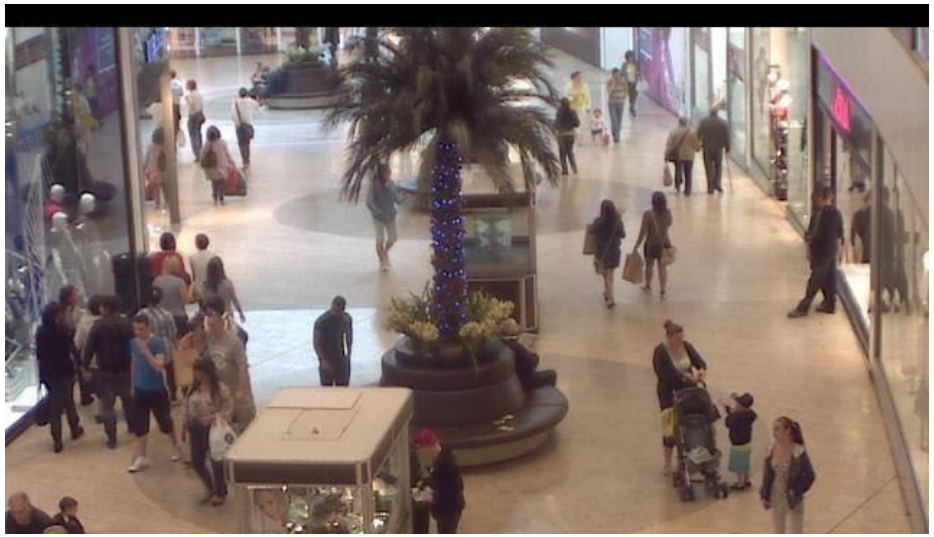
Dataset:

2000 Images

Size:480x640x3

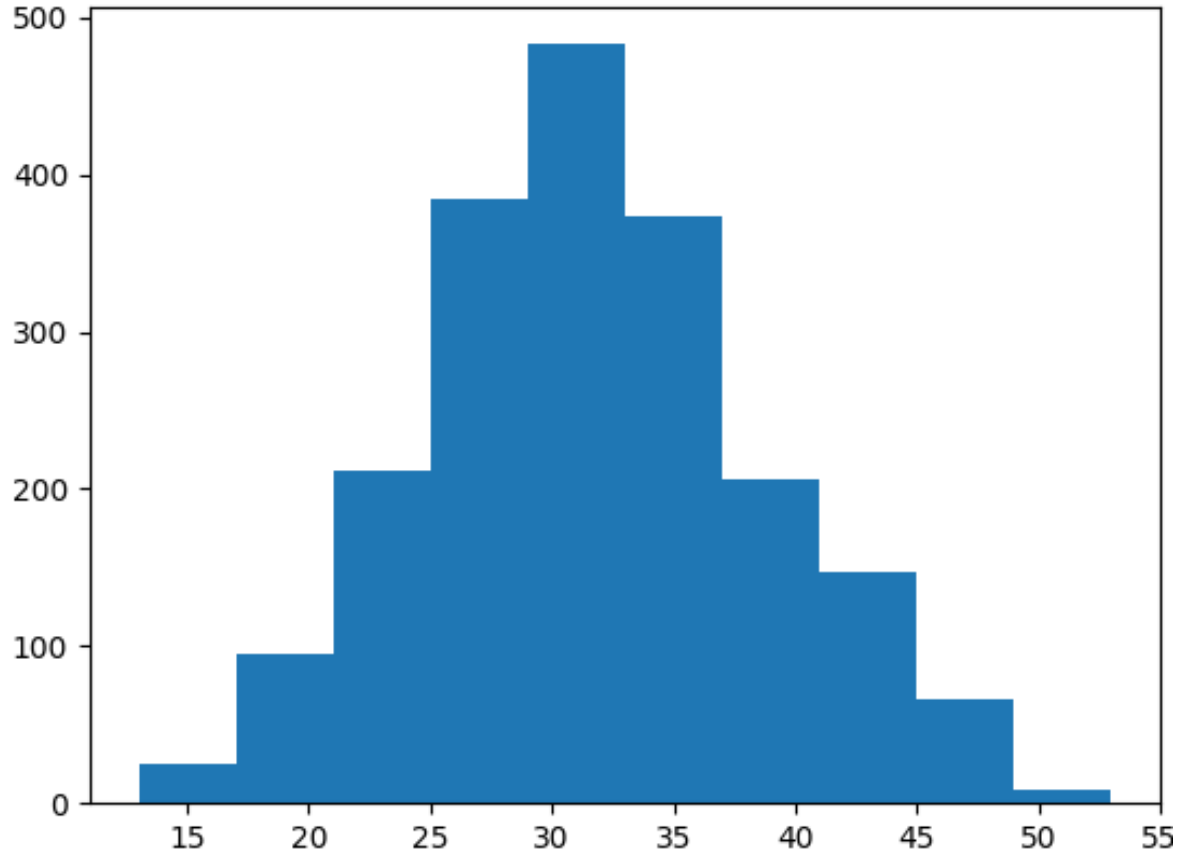
Rezised:128x96x3





Labels:

The data is skewed

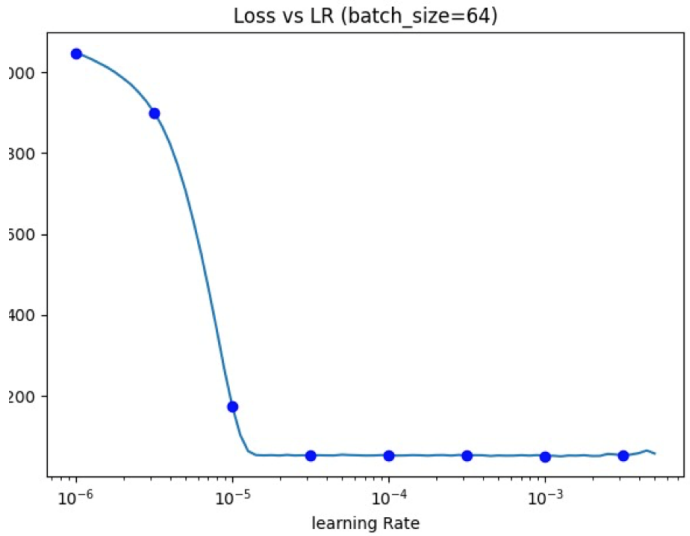


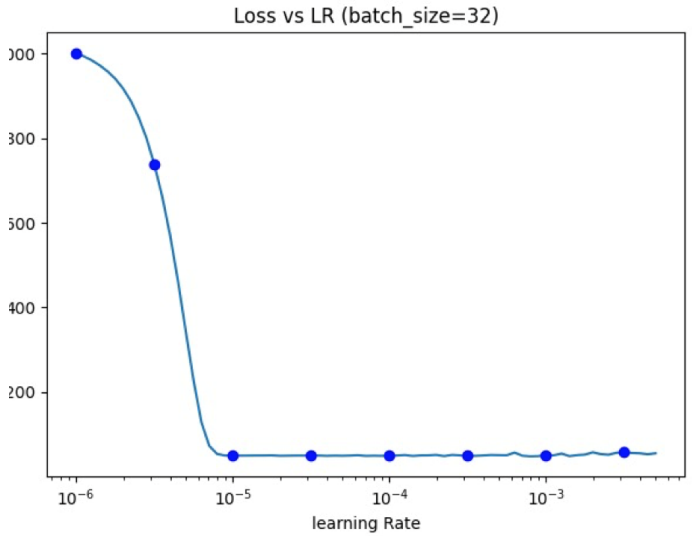
Data Generation and Augmentation

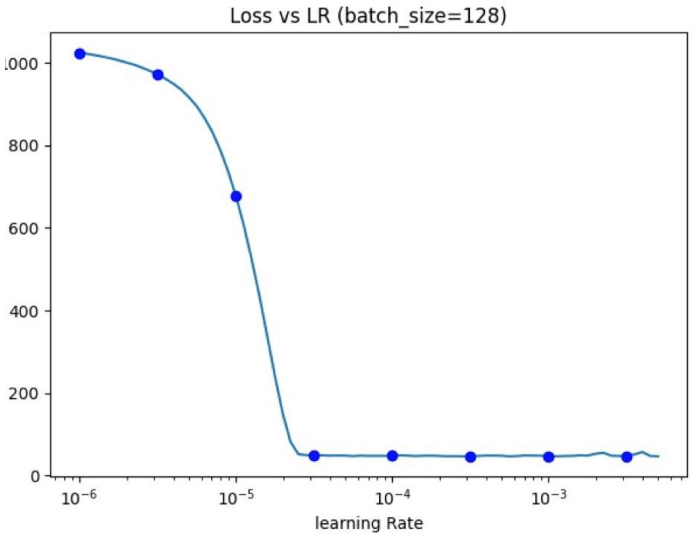
```
53
54  # add ImageDataGenerator
55  datagen = ImageDataGenerator(
56      featurewise_center=False,
57      samplewise_center=False,
58      featurewise_std_normalization=False,
59      samplewise_std_normalization=False,
60      zca_whitening=False,
61      rotation_range=30,
62      width_shift_range=0.1,
63      height_shift_range=0.1,
64      horizontal_flip=True,
65      vertical_flip=False,
66      shear_range=0.5)
67  datagen.fit(x_train)
```

```
61
62  # add a learning rate monitor to get the plot of loss with different learning rate
63  LR_monitor = tf.keras.callbacks.LearningRateScheduler(
64      lambda epochs: 1e-6 * 10 ** (epochs / 20))
65
66  history = model.fit(x_train, y_train, validation_data=(x_test, y_test),
67                      epochs=75, batch_size=32, callbacks=[LR_monitor])
```

Hyper parameter selection- Learning rate







Selecting batch size:

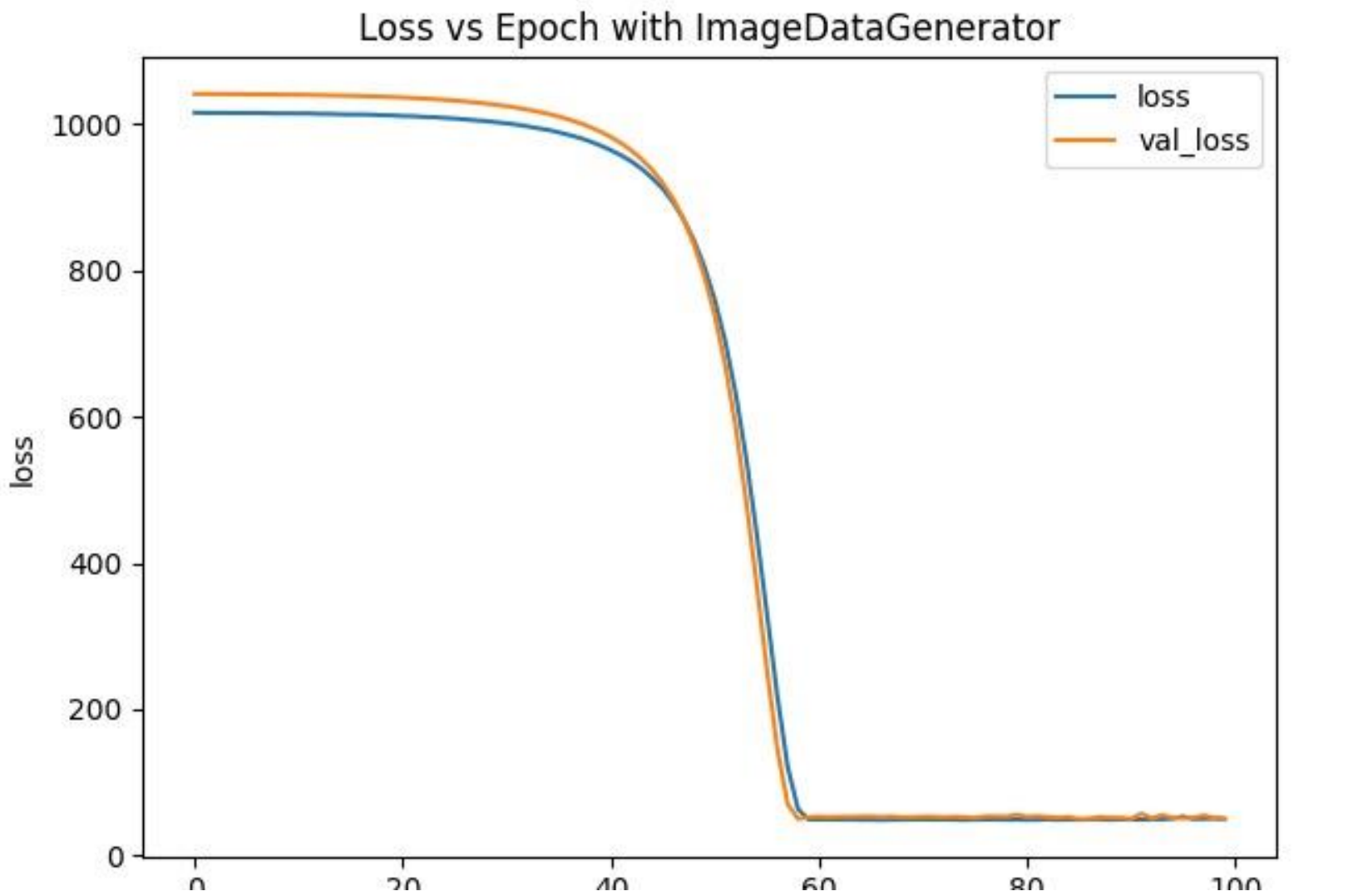
The batch size of 32 is selected

CNN Model

```
# create CNN model
model = tf.keras.Sequential([

    tf.keras.layers.Conv2D(16, (5, 5), input_shape=(128, 96, 3), activation=tf.keras.activations.relu),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation=tf.keras.activations.relu),
    tf.keras.layers.MaxPool2D(2, 2),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(1)
```

- epochs = 100
- Adam optimizer

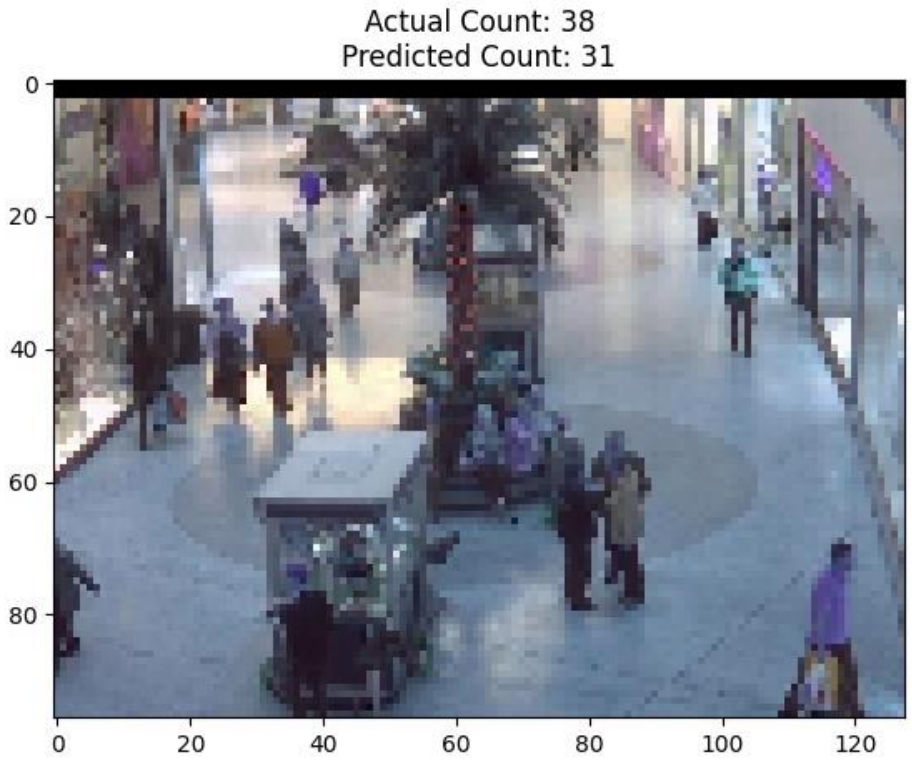
- model_check = ModelCheckpoint("keras-CNN-128-96.hdf5", monitor="val_loss", verbose=1, save_best_only=True)



CNN Model Evaluation:

val_loss: 49.9336

val_mae: 5.4904



CNN Model Predict Result

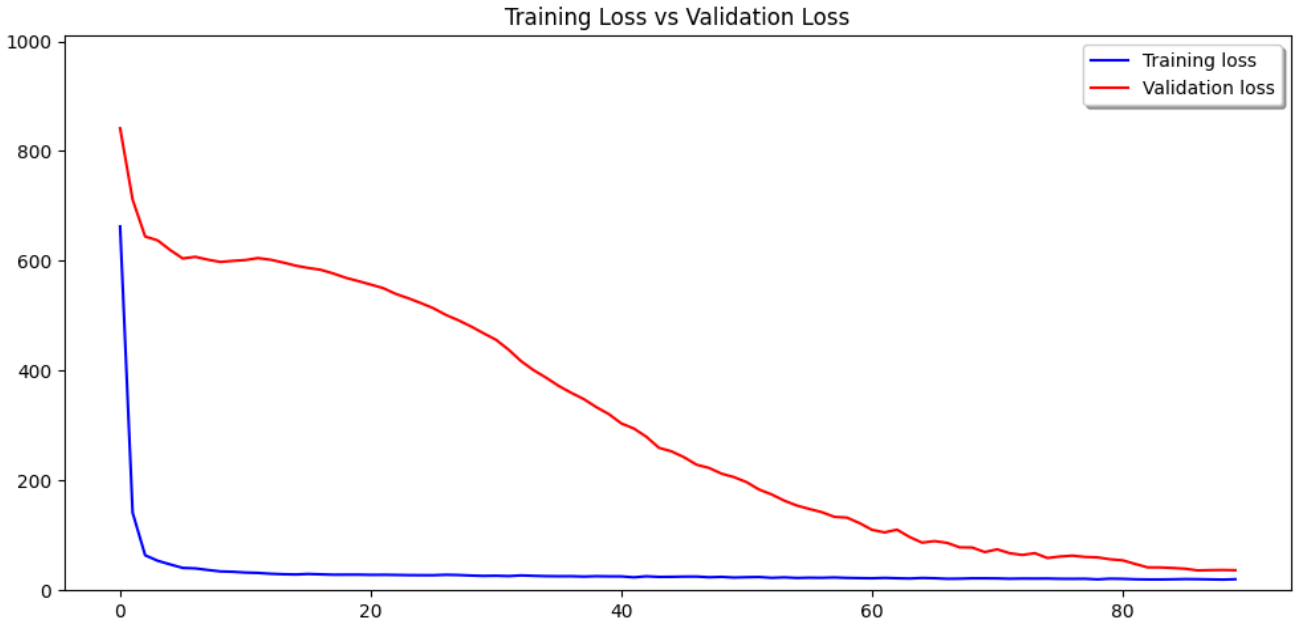
ResNet 50:

- ResNet-50 is a convolutional neural network that is 50 layers deep.
- 90 Epochs were used for training
- Made last 7 layers untrainable
  - Modify the output layer to predict the number of people in the image
  - Prevent Overfitting
- Adam optimizer used for training
- ReduceLROnPlateau()
  - Monitor on the validation mean square error
  - Learning rate drop 20% every 3 epochs, if the MAE does not go down
  - Lowest learning rate will not go beyond 1e-6

```
learning_rate_reduction = ReduceLROnPlateau(
    monitor='val_mean_absolute_error',
    patience=3,
    verbose=1,
    factor=0.2,
    min_lr=0.000001
)
```

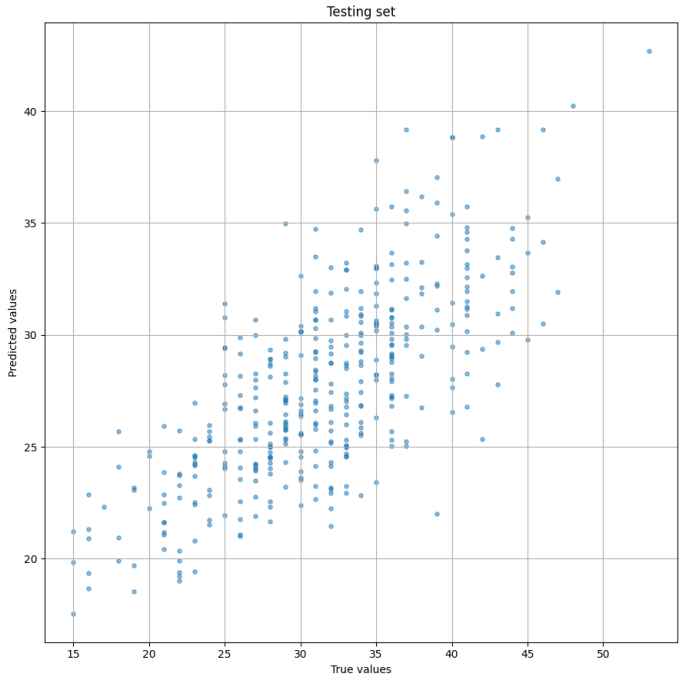
```
optimizer = Adam(learning_rate=3e-4)

model.compile(
    optimizer=optimizer,
    loss="mean_squared_error",
    metrics=['mean_absolute_error']
)
```

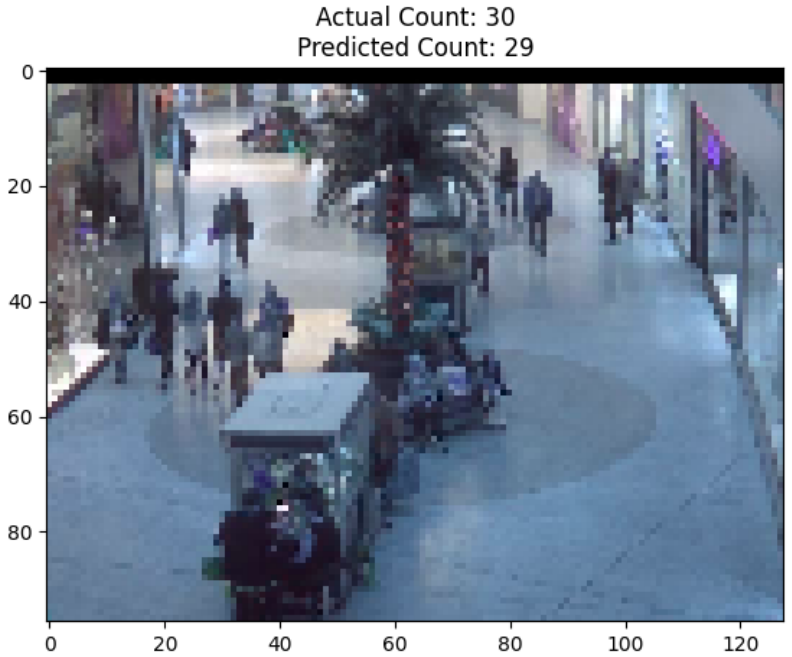
```
MSE Train Set Loss value 20.26082992553711
MSE Validation Set Loss value 36.21799850463867
MAE Train Set Loss value 3.5935373306274414
MAE Validation Set Loss value 4.916306972503662
```



ResNet 50- Results

ResNet 50- Evaluation





Conclusion:

- The ResNet50 model performs better

Models	Mean Absolute Error	Mean Square Error
CNN	5.55	50.29
ResNet 50	4.92	36.31

Future Scope

- MCNN model
- Attempt to decrease loss value by ensembling models
- Find more relevant datasets

Thank you!