Progress Report on CNN Model Development

Initial Model:

Model Used: EfficientNetB0

```
base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(len(top10dict), activation='softmax')(x)
```

Accuracy Achieved: 17.5%

First Custom CNN Model:

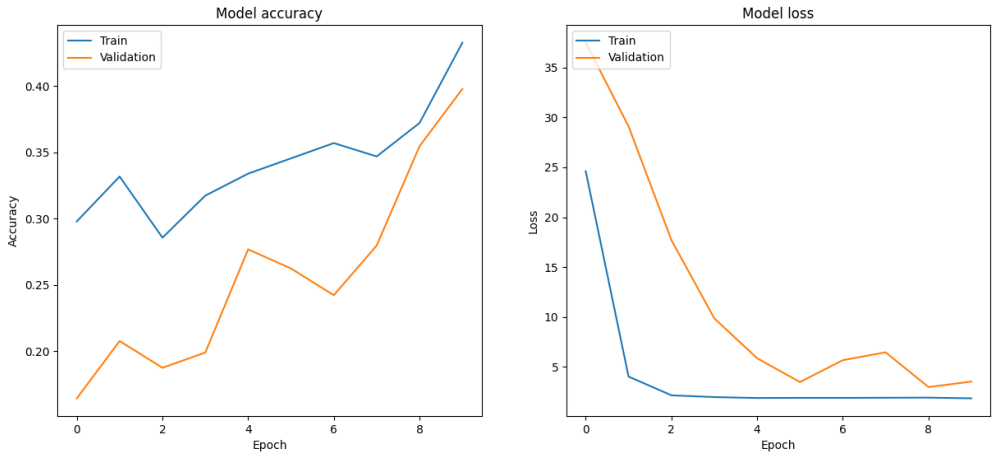
```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(top10dict), activation='softmax'))
```

Accuracy Achieved: 40%

Second Custom CNN Model:

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(top10dict), activation='softmax'))
```

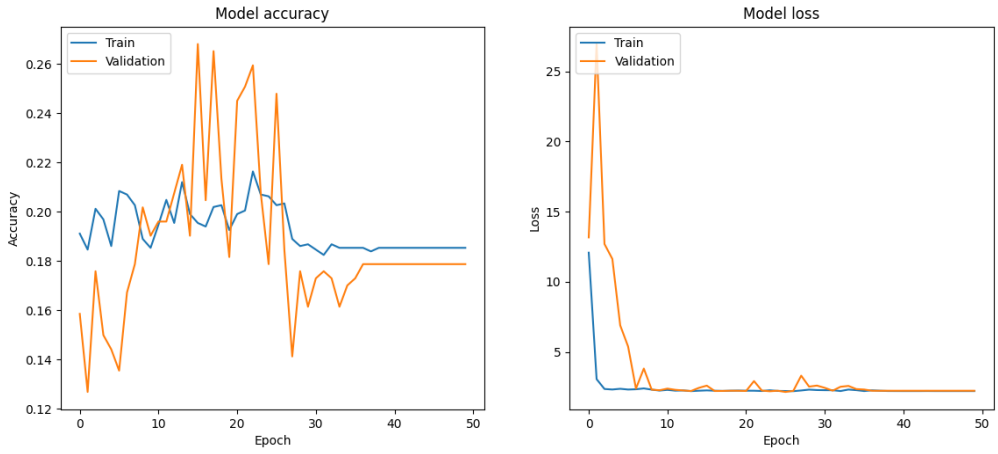
Accuracy Achieved: 39%



Third Custom CNN Model:

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(top10dict), activation='softmax'))
```

Accuracy Achieved: 17%

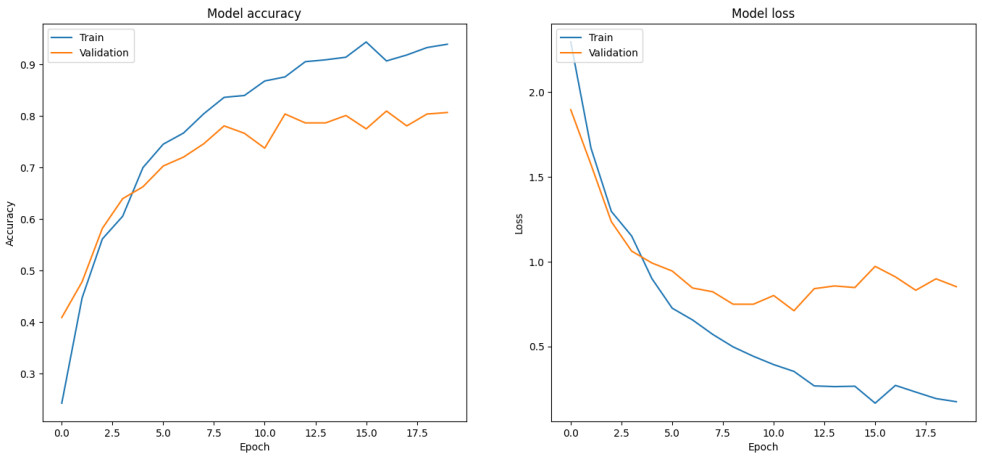


Best Performing Custom CNN Model:

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(top10dict), activation='softmax'))
```

Training Epochs: 30

Accuracy Achieved: 83%



Summary:

The initial EfficientNetB0 model gave a baseline accuracy of 17.5%.

Various custom CNN models were experimented with, achieving up to 40% accuracy.

The best performing model achieved 83% accuracy after 30 epochs by simplifying the architecture and removing batch normalization layers.

Future Work:

Experiment with different learning rates, batch sizes, and optimizers to further enhance model performance.

Combine the predictions of multiple models to improve accuracy and robustness. (Ensemble Methods)

Experiment with more complex architectures, including deeper networks and different types of layers.

Evaluation Metrics: Use additional evaluation metrics (e.g., precision, recall, F1 score) to gain a more comprehensive understanding of model performance.