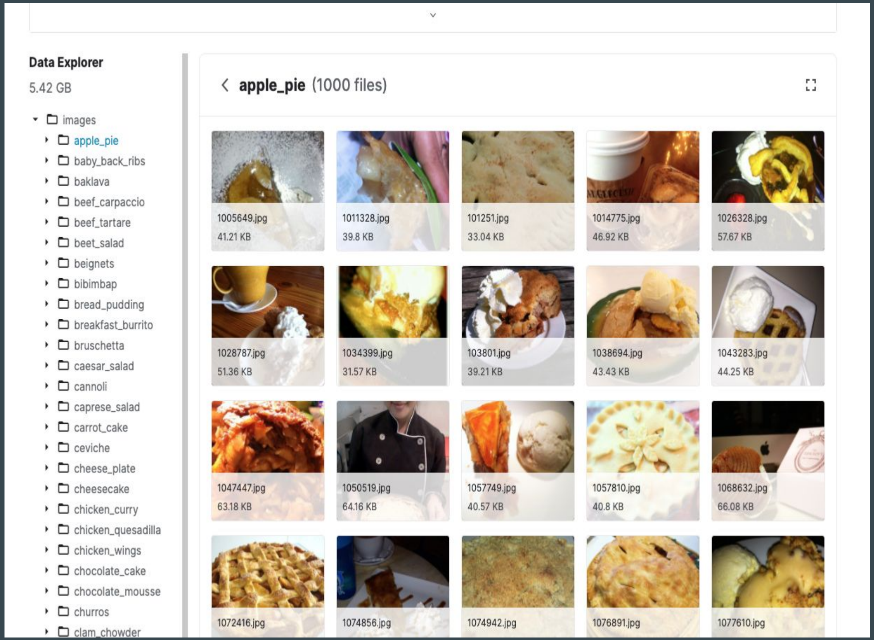
Food Image Classification and Model Analysis

Ty Farris, Sean Nesbit, Marine Cossoul

Overview

- Goal:
  Implement a classification model to predict the category based on a food image

- Food 101 Dataset
  - 101 classes of food
  - 1000 images per category



Problem: Training taking too long

- ❖ 101 classes totaling 101,000 images is a lot to process
- ❖ Google Colab is not fast enough to read the data out of the drives
  - ➢ Used Keras ImageDataGenerator
- ❖ Realized we weren't utilizing the GPU

Solution:

- ❖ Preprocess the data using OpenCV and NumPy
  - ➢ Save it back to the drive
- ❖ Took up more space on the drive than we had available
  - ➢ Could not process all 101 classes
  - ➢ Or we could take ~200 images of each class

Problem: RAM was overflowing

- ❖ Even with 200 images per class, we could not load all 101 classes
- ❖ RAM would overflow and crash the notebook
- ❖ Lose progress on that entire run

Solution:

- ❖ Cut down images to ~70 images/class
- ❖ Allowed us to train the model using the prebuilt NumPy files in a seconds

Problem: Low accuracies

- ❖ Decreased number of images to 70/class
- ❖ Led to low overall accuracies and significant overfitting

Solution:

- ❖ Decrease the number of classes for training
  - ➢ 101 ⇨ 20
- ❖ Increase the number of images per class
  - ➢ 70 ⇨ 200

Model Architecture Revisions

- Dropout
- Regularizers
- Convolutional layers
- Train/Test/Validation Split

```python
import tensorflow as tf
cnn = tf.keras.models.Sequential()
cnn.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, activation='relu', input_shape=[128, 128, 3]))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=128, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=128, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.Dropout(0.25))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Flatten())
cnn.add(tf.keras.layers.Dense(128, activation='relu', kernel_regularizer =tf.keras.regularizers.l2( l=0.01)))
cnn.add(tf.keras.layers.Dropout(0.25))
cnn.add(tf.keras.layers.Dense(len(labels), activation='softmax'))
cnn.summary()
```

Pivoting Back

- When researching architectures for the Food-101 dataset:
  - VGG architectures rarely got above 70%
  - Elite Deep Learning and SENet reached above 80%
  - We did not have the skills to develop those more elite approaches

- After Project Status Meeting, decided to switch from NumPy back to Keras ImageDataGenerator

- Wanted to instead analyze the difference in validation accuracy and training time between various types of image augmentations
  - Sheer/Zoom/Flip, + Rotations, + Brightness, + Grayscale

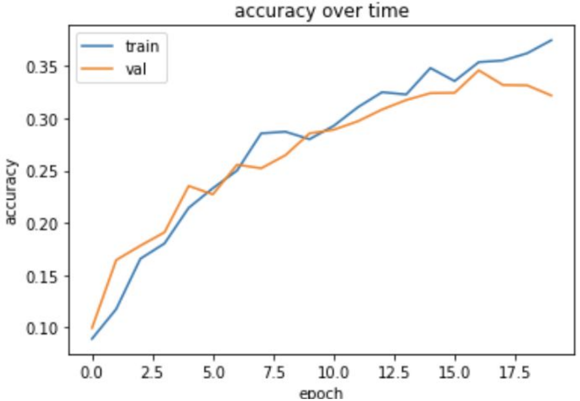
Data Augmentation

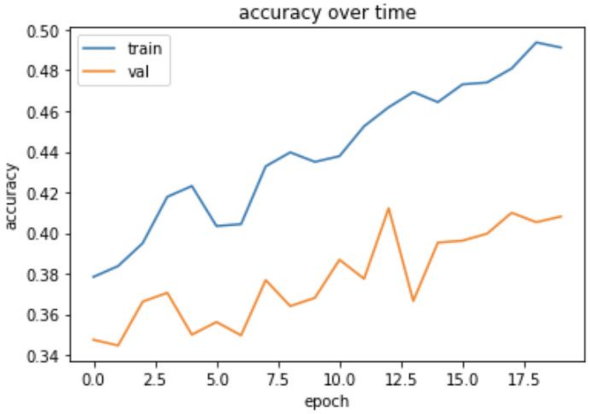
- ❖ Standard (128x128 + Sheer + Zoom + Flip)
- ❖ Gray scaling
- ❖ Rotation
- ❖ Brightness
- ❖ Image Size (64x64)

```
import tensorflow as tf
tf.keras.backend.clear_session()
num_classes = 20

cnn = tf.keras.models.Sequential()
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=[img_height, img_width, 3]))
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3), activation='relu'))
cnn.add(tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3), activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Flatten())
cnn.add(tf.keras.layers.Dense(128, activation='relu'))
cnn.add(tf.keras.layers.Dense(num_classes, activation='softmax'))

cnn.summary()
cnn.compile(optimizer='adam', loss="categorical_crossentropy", metrics=["accuracy"])
```

Results

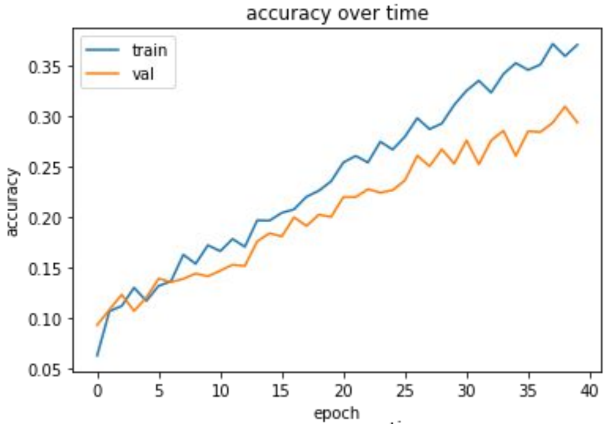
Standard

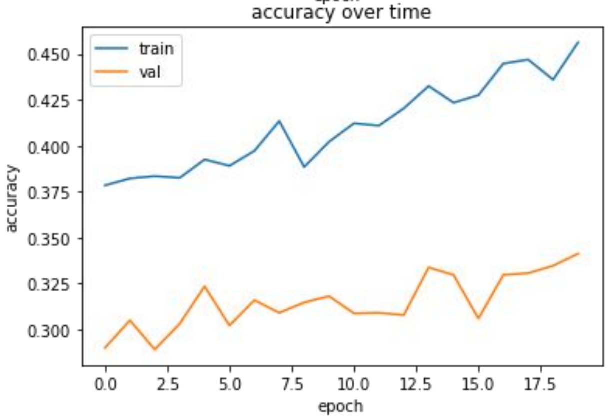




40 Epochs: 49% Train | 40% Validation

Gray Scale

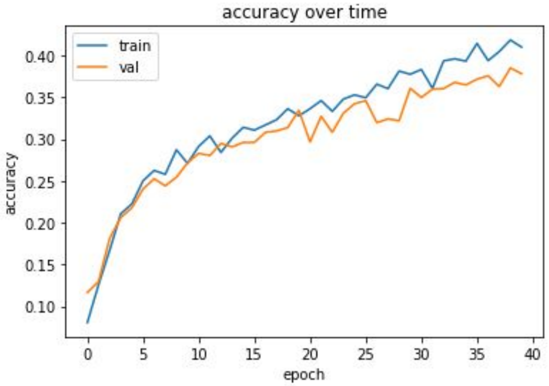


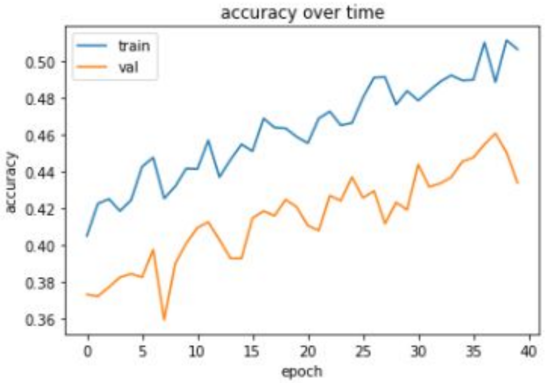


60 Epochs: 46% Train | 34% Validation

Results

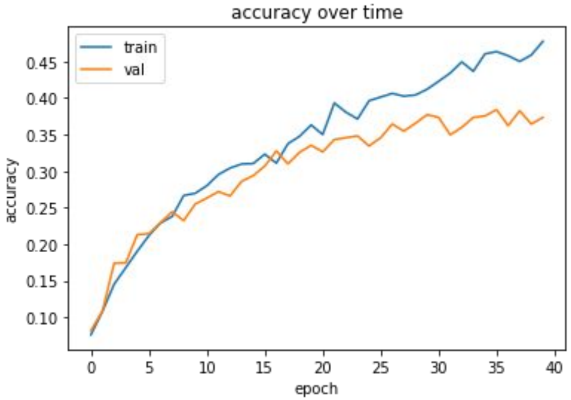
Rotation

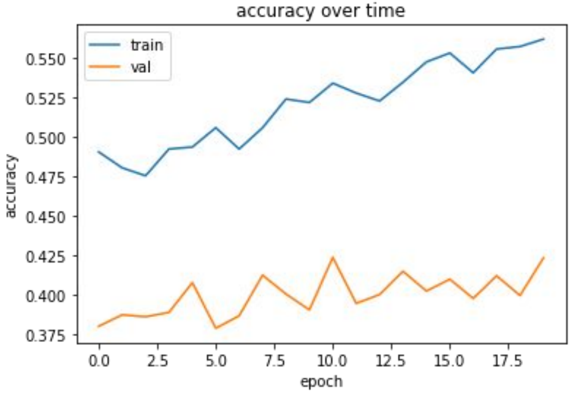




80 Epochs: 50% Train | 43% Validation

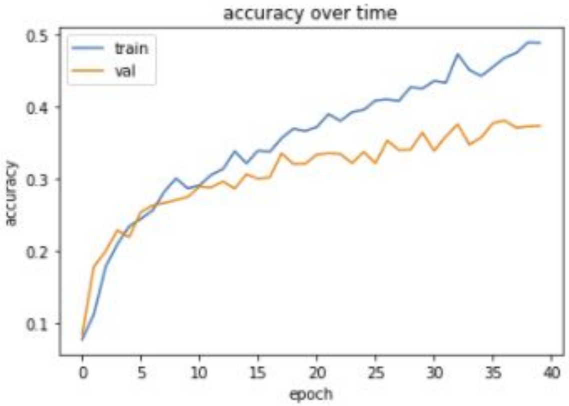
Brightness





60 Epochs: 56% Train | 42% Validation

Image Size (64x64)



60 Epochs: 42% Train | 34% Validation

Discussion of Results

- ❖ We display the epochs where overfitting begins to occur and use those accuracies as our results
- ❖ Rotation and Brightness augmentations have the most significant impact on increasing generalization and validation accuracy
- ❖ More rotations cause longer training and more epochs as it multiplies data size
- ❖ Decreasing image size from 128x128 to 64x64 has a negative effect on validation accuracy, but reduces number of epochs and training time per epoch
- ❖ Grayscale decreases the learning rate significantly, but key features are lost which reduce overall training and validation accuracy

Future Steps

- ❖ Add more data
- ❖ Choose a new dataset
- ❖ Improve model architecture