Classifying Food Images using Convolutional Neural Networks

Ken G Zeng (kgz1), Andrew W Barnett (awb3)
Team name: comp540_kgz1_awb3

Background

Automatic food classification can be used to develop applications that help people monitor their food intake and lead healthier lives. However, this task is challenging problem due to the shear variety of different food groups and the similarity between groups.

Aim

We will be working with a dataset of food related images from the iFood Kaggle competition at FGVC6. This dataset consists of:

- 251 distinct classes
- 118475 training examples,
- 11994 validation images.
- 28377 unlabeled testing examples

Our goal is to train a deep learning model that can classify noisy images of food from this dataset. The model will be scored based on the top 3 accuracy on the testing set.

Exploratory Data Analysis

Figure 1.1 Arithmetic mean of the first five classes.

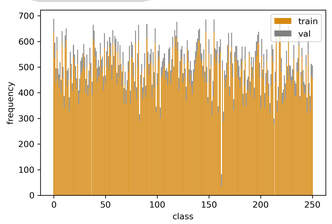

Figure 1.2. The number of training and validation examples for each class.

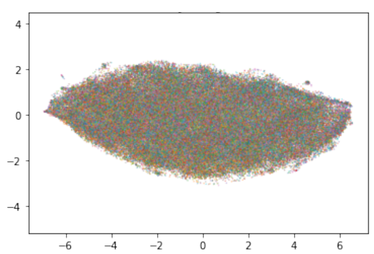

Figure 1.3. Reducing the dimension of each image vector to 2D. The colours for each label is generated randomly

Preprocessing

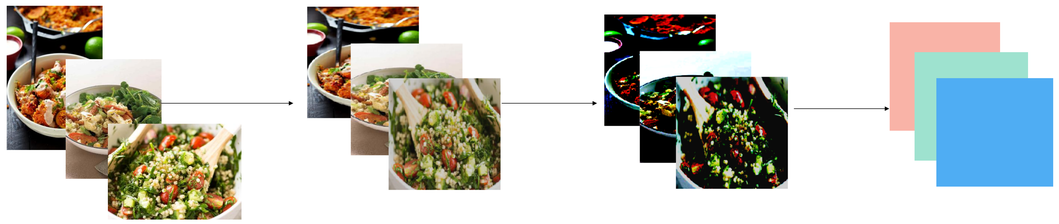

Figure 2.1 The input contains images of varying sizes.

Figure 2,2 The images are resized. Then augmented by cropping or flipping horizontally.

Figure 2,3 The are converted into tensors and then normalized.

Figure 2,4 loaded onto the GPU.

Model Selection

Initially we worked with three main structures: VGG11, ResNet101 and RestNet152. We trained each of these for two epoch with the following hyperparameters:

- Weights: pretrained
- Learning rate: 0.01
- Batch size:16.
- Optimizer: SGD
- momentum of 0.9
- Loss function: Cross entropy

Giving us the following results:

	Parameters	Validation loss	Validation accuracy	Kaggle Score
VGG11	129800187	4.71	25.45	0.43650
ResNet101	43014459	2.57	51.35	0.28462
ResNet152	58658107	1.49	53.13	0.27311

We ended up choosing ResNet152 to fine tune because it achieved the highest score after two epochs.

Figure 3.2. Feature visualization of class 0, the category for the macaron, using Lucid.

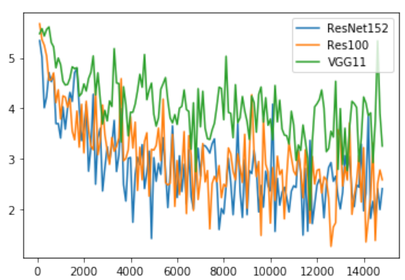

Figure 3.2. The change in loss value after each 100 iteration for the three models

Parameter Tuning

We then trained the final model by training for two epochs at a time, saving the network, then tuning the starting learning rate..

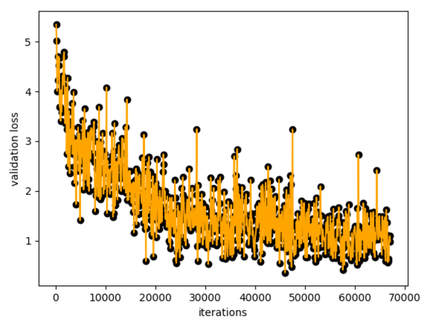

Fig. 4.1. The output of the loss function after each 100 iteration.

We used the training loss curve to optimize the hyperparameters. If there's little progress, we increase the learning rate by a factor of 2-5. If the loss curve becomes too volatile, we reduce the learning rate. We repeated this for 8 epochs to see if we can further reduce the validation loss.

Results

Figure 4.1. Feature visualization for the first 5 classes using Lucid.

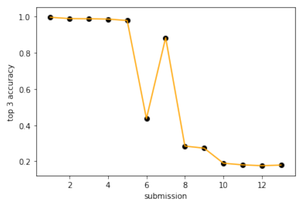

Fig 4.2. Kaggle score after each submission. Here's a basic description of the method for each one:

1-5: faulty implementation
6-7: VGG11
8: ResNet101
9 – 13: ResNet152

The model accuracy did not improve after the 5th epoch hence that's the weights at that point will be our final model. This model achieved a score of 0.17643 on Kaggle.

Future Considerations

- It might be possible to consider methods that work on images of different sizes
- different set of weight values.
- We also wanted to test out other popular neural net structures such as Inception.
- Finally, an ensemble method using the outputs of multiple different models may help us reach a better score

Acknowledgements

1. torch.utils.data¶. torch.utils.data - PyTorch master documentation. https://pytorch.org/docs/stable/data.html. Accessed April 24, 2020.
2. Elichen. elichen/Feature-visualization. GitHub. https://github.com/elichen/Feature-visualization/blob/master/Feature visualization.ipynb. Accessed April 24, 2020.
3. torchvision.models¶. torchvision.models - PyTorch master documentation. https://pytorch.org/docs/stable/torchvision/models.html. Accessed April 24, 2020.
4. Aramis, Rosebrock A, Ruben, et al. ImageNet: VGGNet, ResNet, Inception, and Xception with Keras. PyImageSearch. https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/. Published April 18, 2020. Accessed April 24, 2020.
5. UMAP for Supervised Dimension Reduction and Metric Learning¶. UMAP for Supervised Dimension Reduction and Metric Learning - umap 0.4 documentation. https://umap-learn.readthedocs.io/en/latest/supervised.html. Accessed April 24, 2020.
6. Mee J, Mee JMJ. How to generate random colors in matplotlib? Stack Overflow. https://stackoverflow.com/questions/14720331/how-to-generate-random-colors-in-matplotlib. Published November 1, 1962. Accessed April 24, 2020.\\
7. Peixeiro M. Hitchhiker's Guide to Residual Networks (ResNet) in Keras. Medium. https://towardsdatascience.com/hitchhikers-guide-to-residual-networks-resnet-in-keras-385ec01ec8ff. Published April 8, 2019. Accessed April 24, 2020.