SeeFood: 10 class dessert classifier

Kevin Wee, DSI20

Problem Statement

Currently, unmanned supermarkets/ convenience stores are gaining traction in China. Given the current pandemic situation, such a business model is ideal, as it minimizes human contact, and reduces queues.

Can we create a food classifier to facilitate self checkouts, so as to help retail and F&B businesses save on manpower and productivity costs?

METHODOLOGY

EDA

Exploring files and folders in dataset

Visualise images

Check for imbalanced classes

Experiment

Created 5 class Japanese food classifier

Transfer Learning on VGG16

Val accuracy 84.8%

Modelling and Evaluation

Created 10 class desserts dataset

Trained dataset on:
Custom CNN
VGG16
EfficientNetB0

Tuned models for retraining

Best model: Tuned EfficientNetB0 (Val accuracy: 77.6%)

Analysis on misclassified images

Created confusion matrix

Visualised misclassified images

Most commonly misclassified classes:
Tiramisu
Carrot Cake
Cheesecake

Testing on unseen data

Test on googled images

Created webapp (deployed)

EDA

DATASET EXPLORATION

- Food-101 dataset from Kaggle

Meta

Files in json, txt format with information about dataset.

Images

101 folders of image data ranging from apple pies to waffles.
1000 images per category of food.

HDF5 files

Reformatted data in Keras HDF5 Matrix format, e.g.
foodc*101*n1000_r384x384x3.h5
(101 categories, 1000 images, 384 x 384 x 3 (RGB, uint8))

INSIDE THE IMAGE FOLDERS

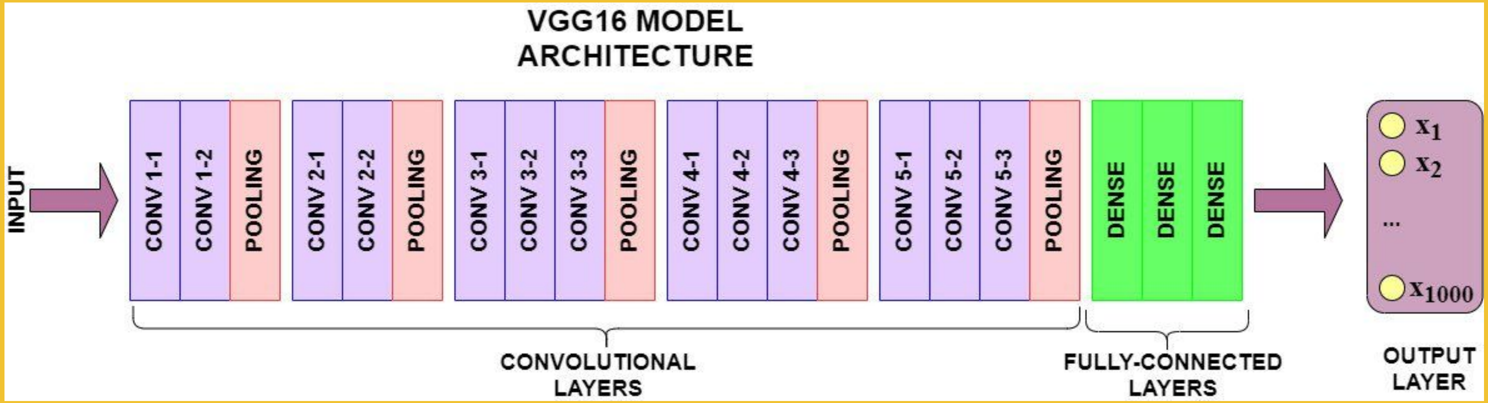
INSIDE THE IMAGE FOLDERS

Experiment

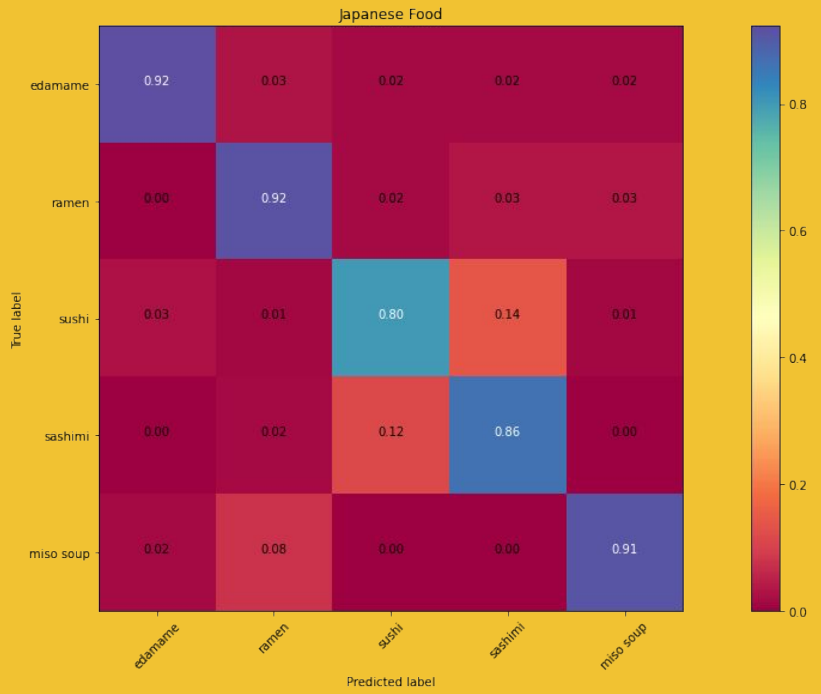
Transfer learning with 5 classes of Japanese Food

JAPANESE FOOD CLASSIFIER (5 CLASSES)

- Transfer learning with VGG16
- VGG16: pretrained convolutional network model (achieved 92.7% top-5 test accuracy in ImageNet)
- 5 classes: Edamame, Ramen, Sushi, Sashimi, Miso Soup
- Freeze all convolutional layers, remove top dense/prediction layers
- Rebuilt top layers with: Global Average Pooling, Dense, Dropout and Prediction



JAPANESE FOOD CLASSIFIER: 84.8% Validation Accuracy



- ❖ Can predict classes with reasonable accuracy
- ❖ Might sometimes misclassify sushi with sashimi and vice versa

```
50/50 [==============================] - 11s 215ms/step - loss: 0.6096 - acc: 0.8487
model accuracy: 0.8487499952316284
```

JAPANESE FOOD CLASSIFIER: Predicting Online Images

sushi

edamame

```
result= np.argmax(pred)
if result==0:
    print("edamame")
elif result==1:
    print("ramen")
elif result==2:
    print("sushi")
elif result==3:
    print("sashimi")
elif result==4:
    print("miso soup")
```

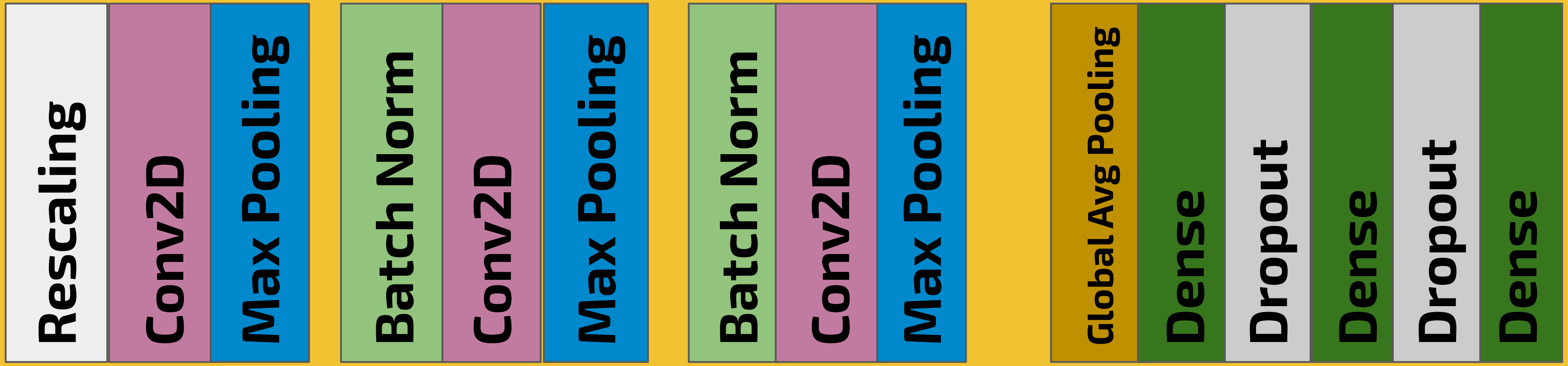
Modelling

10 class dessert classifier

ORIGINAL HYPERPARAMETERS

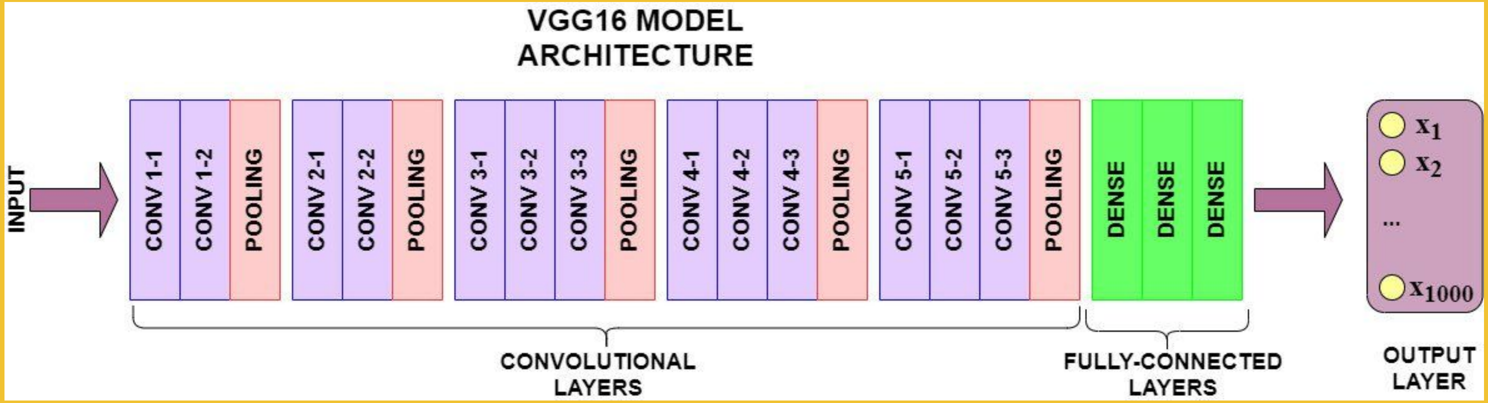
- Batch Size: 32
- Learning Rate: 0.001
- Optimiser: Adam
- Loss: Categorical Crossentopy
- Epochs: 100
- Train-Test-Split: 0.75/0.25
- Metrics monitored: val loss (early stopping after 10 epochs), val accuracy

CUSTOM CNN



- ❖ Retained dimensions for first layer
- ❖ Tuned and re-trained using best hyperparameters from Keras Tuner (optimal units in dense layer: 64, optimal learning rate: 0.0001)
- ❖ Regularization techniques: Batch Normalization, Dropout, Early Stopping

VGG16



- Rebuilt top layers with Global Average Pooling, Batch Normalisation, Dropout, and Dense layers
- Tuned and re-trained by unfreezing block 5
- Regularization techniques: Batch Normalization, Dropout, Early Stopping, lowering learning rate

EfficientNetB0

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

- Rebuilt top layers with Global Average Pooling, Batch Normalisation, Dropout, and Dense layers
- Augmented images before training
- Tuned and re-trained by unfreezing block 7a (BatchNormalization layers remain frozen)
- Regularization techniques: Batch Normalization, Dropout, Early Stopping, lowering learning rate

MODEL EVALUATION

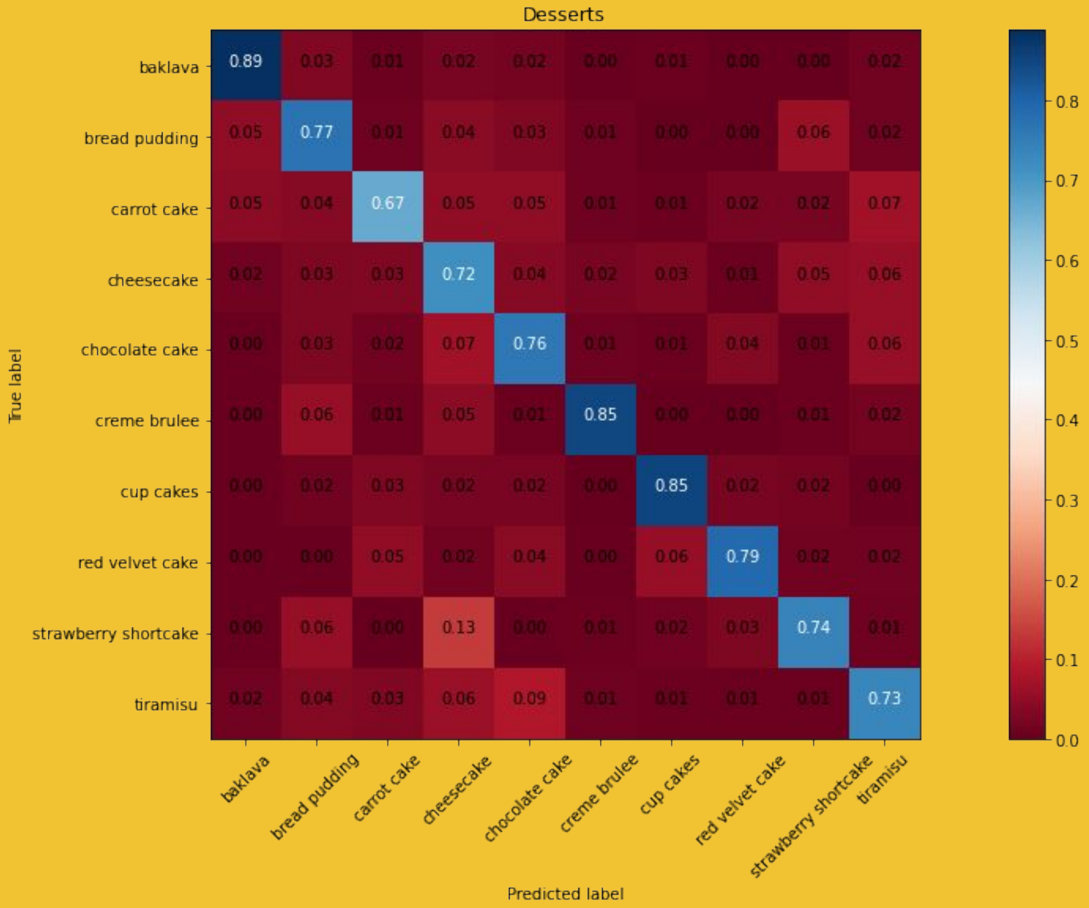
Model	Val Accuracy (before tuning)	Val Accuracy (after tuning)
Custom CNN	46.3%	56.7%
VGG 16	59%	75.6%
EfficientNetB0	73.2%	77.6%

Tuned EfficientNetB0 is our best model, achieving a validation accuracy score of around 77.6%.

Misclassified Images

10 class dessert classifier

CONFUSION MATRIX



MISCLASSIFIED IMAGES

- ❖ Misclassification seems reasonable
- ❖ Lighting and presence of toppings/other ingredients on food might have caused misclassification

Model Testing

10 class dessert classifier

WEBAPP DEMO

10 Class Desserts Classifier

This is a simple image classification webapp to predict 10 classes of desserts: baklava, bread pudding, carrot cake, cheese cake, cupcakes, chocolate cake, tiramisu, red velvet cake, strawberry shortcake and creme brulee. The model was trained on EfficientNetB0 and has achieved 77% validation accuracy.

Do note that the classifier is not 100% accurate and may tend to misclassify certain images like carrot cake with cheesecake etc.

Please upload an image file

Drag and drop file here
Limit 200MB per file • JPG, PNG

Browse files

carrotcake.jpg 95.5KB

It is carrot cake!

https://share.streamlit.io/kevwee-lab/dessertclassifier_webapp/main/foodclass_app.py

Conclusion

10 class dessert classifier

CONCLUSION

- ❖ Managed to build 10 class desserts classifier with val accuracy around 77.6%
- ❖ Room for improvement, as there are still misclassifications

Next Steps/Room for improvement

- ❖ Train model with more data of food at different angles and height
- ❖ Train model with more classes of food
- ❖ Try out other models/ensemble models/ image augmentation e.g. CutMix to improve accuracy
- ❖ Create object detection model to count items detected

LEARNINGS

- Retain image dimensions in your first layer, as the machine is learning the edges and features of your data
- Batch Normalization helps improve your accuracy score and allows your model to converge in lesser epochs. It also provides some regularization and reduces generalization error
- Always push your model's (and your own) limits. Keep training, fine tuning and experimenting to improve val accuracy and minimise val loss

THANK YOU