Machine Learning Engineer Nanodegree

Capstone Project

Mark Wright - February 1st 2018

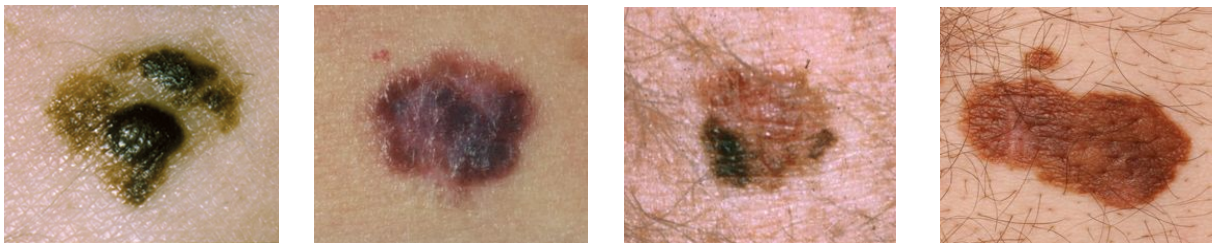
I. Definition

Project Overview

Skin cancer rates globally are rising, according to the the WHO between 2-3 million non melanoma skin cancers and 132,000 melanoma skin cancers (a more dangerous form that is more easily spread to other parts of the body) occur every year, and 1 in every 3 cancer diagnoses are a skin cancer [1]. Incidence rates for melanoma skin cancer are projected to rise by 7% in the UK between 2014 and 2035 [2].

Skin cancers are usually diagnosed by a clinical specialist after a referral by a GP by visually inspecting the potential cancer, or an image of the potential cancer, followed by a biopsy to confirm a classification if they think the specimen likely cancerous. There are several visual features that may indicate that a melanoma is cancerous. For example it could be unsymmetrical, have irregular borders, have uneven color, and/or be of a large size (at least the size of a pencil tip) [3].

Symmetry | Border | Color | Size



Images from Cancer Research UK [2]

Since there are visual clues in the diagnosis of skin cancer it is a suitable subject to utilise machine learning and computer vision to help specialists to make decisions. A classifier could be written that given an image of a specimen would indicate the likelihood of it being cancerous.

Problem Statement

I will make use of deep Convolutional Neural Networks to train a classifier that will be able to identify if an image of a mole is benign or malignant. I will use transfer learning to make use of an existing general image processing model that have already been trained to significantly reduce the training time required. I will remove the final fully connected layers of the model and replace with a new fully connected layer configured to classify images as benign or malignant. I also will experiment to see if an ensemble of trained CNN models, using the technique detailed above, achieves better or worse performance than the individual models.

Metrics

Since this is a medical test, it is far more important that we do not miss patients who have a disease rather than incorrect diagnose someone who does not have the disease. Therefore sensitivity (recall) will be favored over specificity (precision) in measuring the performance of the final model.

$$\text{sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}$$

Image from https://uberpython.wordpress.com/

$$\text{specificity} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}}$$

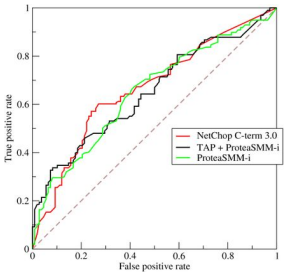
Image from https://uberpython.wordpress.com/

I will also use the f1 score which balances precision/recall.

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Image from https://en.wikipedia.org/wiki/F1_score

I will calculate a ROC curve [5] for the final model produced, to show the true positive vs. false positive rates at various decision thresholds.



Example ROC Curve Image from wikipedia[5]

II. Analysis

Data Exploration

The International Skin Imaging Collaboration: Melanoma Project have accumulated a large labelled image archive of skin cancer images that are publicly available for teaching and the testing of automated diagnostic systems [7]. With the aim to reduce melanoma mortality.

There are a total of 13,786 images in the archive [8]. 12668 are classified as benign, and 1084 are malignant. The archive is accompanied by a gallery navigation tool (powered by girder [9]) which allows for the images to be browsed and filtered by various facets. Each image is paired with a json file which stores metadata about the image including features such as whether the specimen was found to be benign/malignant, some data about the patient such as sex, age and family history. There are many features within the metadata for images which is not recorded for each image. I have decided for the course of this project to ignore all metadata apart from the target variable ‘benign_malignant’.

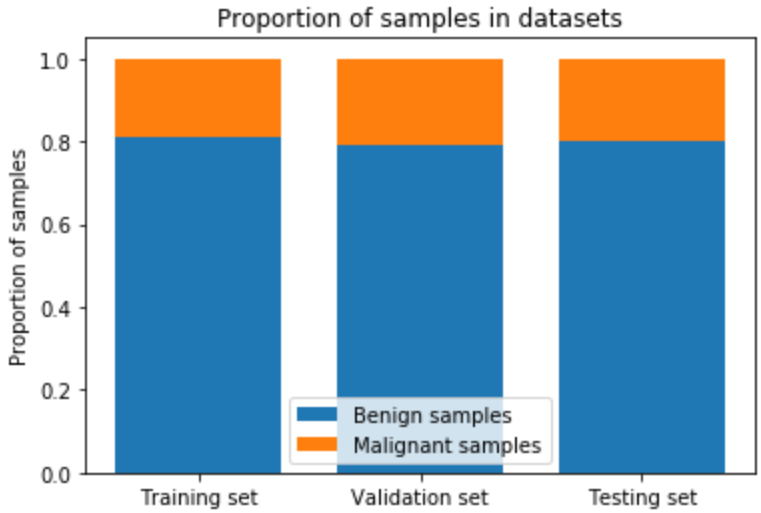
I will utilise a subset of the dataset that was partitioned for the 2017 ISBI challenge. It contains a total of 2000 training images, 150 validation images and 600 testing images. [10] I will maintain these partitions for my project.

One important thing that I wanted to make sure was that the proportion of samples in each of the datasets were similar, so that this did not bias the accuracy of my results.

Proportion of benign to malignant samples in training set is 0.81% to 0.19%
Proportion of benign to malignant samples in validation set is 0.7933333333333333% to 0.20666666666666667%
Proportion of benign to malignant samples in test set is 0.8016666666666666% to 0.19833333333333333%

Due to the distribution of different sample classes I will need to ensure that this is addressed by my model or it may lead to bias.

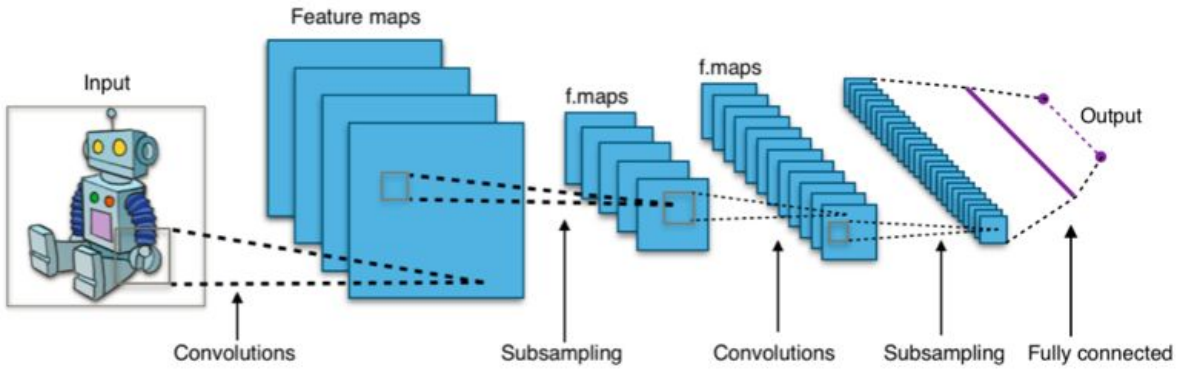
Exploratory Visualization



This is a plot that shows the proportion of samples in each of the datasets I will use to train, validate and test the accuracy of my model, I wanted to make sure that they had similar proportions so that there was less chance of bias.

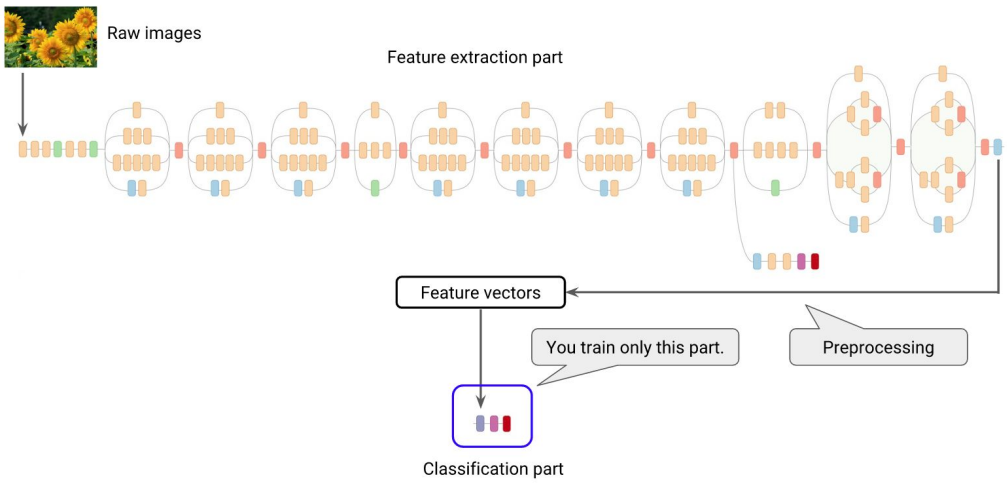
Algorithms and Techniques

I will use convolutional neural networks to train a classifier to predict whether an image is benign or malignant. Convolutional neural networks (CNN's) are a class of deep neural networks that have successfully been applied to analysing visual imagery. They do this by mimicking connectivity patterns in the neurons in the visual cortex of animals [11].



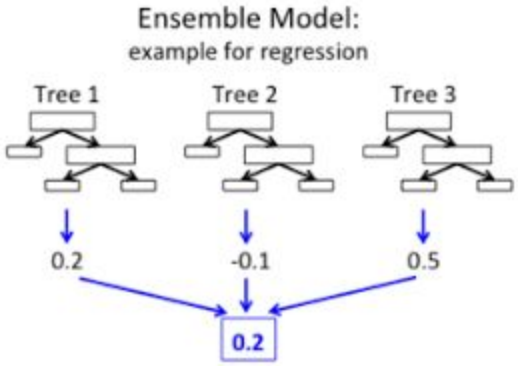
https://en.wikipedia.org/wiki/Convolutional_neural_network

Since the amount of training data available is relatively small I intend to use the process of transfer learning. This is where existing models that have already been trained using significantly more training data are used as the basis of a final classifier. I intend to make use of the following pre built models, Xception [12], ResNet50 [13] and InceptionV3 [14]. For each in turn I will remove their final fully connected classification layer, and add a new fully connected later for the two categories we want to predict (benign/malignant), and then conduct fine tuning of the models.



https://codelabs.developers.google.com/codelabs/cpb102-txf-learning/index.html

Finally I will create a feature vector for each of the 3 trained classifiers and try to use a random forest ensemble method to produce a final prediction to see if this performs better than the individual classifiers.



http://www.nosimpler.me/random-forest/

Benchmark

Last year Esteva et al, trained a CNN using transfer learning that is able to perform at a similar level to dermatologists [15]. For melanomas the average dermatologist in their trial classified around 95% of malignant lesions, and 76% of benign moles correctly. By comparison their algorithm was capable of classifying 96% of malignant lesions and 90% of benign lesions.

Whilst I am unlikely to be able to get anywhere close to this level of accuracy, given the size of my dataset and computational power I have access to, what I will hope to be able to show it that, for a novice data scientist with access to a laptop and open source machine learning libraries such as keras [16], tensorflow [17] and scikit learn [18], that a classifier that operates at a comparable level to a human is possible.

III. Methodology

Data Preprocessing

I downloaded the 2017 ISBI challenge pre generated datasets onto my computer. Each set of images along with their metadata were stored in a folder (train/test/validation), and in each folder there were additional sub folders relating to which collection hosted on the the The International Skin Imaging Collaboration: Melanoma Project.

I created a file directory walker to store the file paths for each image and extract the target variable (malignant/benign) from the json metadata associated with the file. I stored the data representing the image file locations in 3 seperate objects train_files, test_files, validation_files and the target variables in train_targets, test_targets and validation_targets.

I then wrote a function to process each file in each of the image file locations, and generate an input tensor with the correct size and dimensions so that they could be used in the training and testing of my different CNN's, and stored in 3 objects original_train, original_test and original_validation scaling appropriately.

I generated bottleneck features for each of the pre-defined models (Xception, ResNet50, InceptionV3) I was going to use for transfer learning. These were created by feeding the input tensors representing images in the training dataset, through each of the pre-defined models, excluding the existing top classification layer, and outputting the resulting vectors. These vectors would be used as input to my top-layer classifier for each of the models.

To make it easier to re-use data in separate evolutions of my algorithm I decided to write the values for each input set (original_training, original_test and original_validation), and the bottleneck_features (Xception_train/test/validation, resnet_train/test/validation, inception_train/test/validation) that I generated using each model, to disk, this meant that I didn't have to run the computationally expensive process of generating these values from raw images before each evolution of my algorithm.

Implementation

I began by training a model based on the Xception [12] image classification model. Due to the fact that the clinical images I wanted to classify were different from those In the original image net dataset, and the number of training images I had was relatively small I decided to use transfer learning and then fine tuning, to try and improve the classifier.

For the initial learning step I used the bottleneck_Xception dataset as input, flattened the input, added a dropout step (which I varied in several runs to try and prevent overfitting) and added a fully connected layer with 2 outputs 1 for each of the classes (benign/malignant).

I used checkpointing so that the best performing (with the lowest loss when the model was tested using the validation dataset after each epoch) weights as used in the model were stored to disk. I used binary_crossentropy as the method for calculating loss in each iteration of training and RMSProp as an optimiser. I decided to use 200 epochs during the training phase.

I generated a metric for f1_score so that it could be used during training to monitor training progress since it would be much better than the accuracy metric to judge performance because of the unbalanced class distribution.

At first I found that by using the default class_weights for each input class the model would very quickly reach an accuracy of 80%, but this was due to it just classifying every image as benign (since 80% of the images were benign!) so I applied a class_weight factor of 1:4 so that when calculating the model loss, the malignant samples weights were boosted.

Once the model was trained I then used it to predict the classes of the testing dataset and evaluated its performance by calculating the f1_score and by looking at the confusion matrix generated. I decided to use these to assess the model because of the in-balance of the input datasets (80/20) and the importance of predicting malignant over benign (recall/sensitivity).

Once I came up with a suitable base model I then began the process of fine tuning. To do this I loaded the layers of the pretrained Xception model and froze the default weights of the first 116 layers (so they wouldn't be updated during the training process), leaving the top 16 layers. I then connected my fully connected classifier layers on top of them, and set the weights to the optimum weights I calculated during the initial training phase.

I once again used checkpointing so that the optimum model weights were stored to disk, set the class_weight to 4:1, and binary_crossentropy to calculate loss, and the f1_metric to assess performance. I changed the optimizer to SGD with a very small learning rate so not to ruin the weights learnt by the original model. I ran it over 100 epochs, and since more computation occurred with training more layers of the model it took significantly longer to run (over 24 hours on my laptop). I then used the test dataset to once again assess the performance using f1_score and a confusion matrix.

I then repeated this process for the ResNet50 and InceptionV3 models using the bottleneck feature for each model as inputs and the same configuration as above. For ResNet50 in the fine tuning process I tried freezing the weights of layers upto the 4th and then upto the 5th layers. For InceptionV3 I chose to train the top 2 inception blocks as suggested by Keras documentation on fine tuning InceptionV3, therefore I froze the weights of the first 249 layers, and unfroze the rest [17].

Once I had finished creating these CNN classifiers, I stored each of their prediction vectors to disk, including the intermediate models before fine tuning, for the test,validation and training original datasets. I then re formatted the CNN prediction output so that I had 3 features as a new input vector for training (output_xception_malignant, output_resnet_malignant, output_inception_malignant) for each training image. I also created another training set including the intermediate model results before fine tuning in case this performed better: (output_xception_malignant, output_resnet_malignant, output_inception_malignant, output_intermediate_xception_malignant, output_intermediate_inception_malignant).

I decided to concatenate the values for both the training and validation data sets to give me more training data. I also created a test dataset consisting of the same feature vectors from the predictions generated from the original testing dataset.

I then evaluated several methods to attempt to combine the results of each of the individual classifiers evaluating which performed best, Random Forest, AdaBoost, Bagging and an SVC. When attempting to find the best hyper parameters for the SVC I used GridSearch to speed up the process.

Refinement

Xception initial model training

I found that during initial training that when using default class_weights the classifier would bias towards the benign class (since this made up 80% of the data) leading to 80% accuracy by just classifying everything as benign. I tried out using different values of class_weight to boost the malignant class when calculating loss to try and overcome this. I used various different configurations from 1:4, 1:10, 1:20, until settling on 1:4 which evenly represented the distribution of classes in the datasets.

On several evolution of the model I tried several things.

My first model I flattened the input, used a dropout layer with 0.5 and then a fully connected layer with output 2 this received the following accuracy and f1_score:
72% accuracy f1-0.37

My second model I flattened the input, used a dropout layer with 0.9 and then a fully connected layer with output 2 this received the following accuracy and f1_score:
56% accuracy f1-0.29

My third model I flattened the input, used a dropout layer with 0.7 and then a fully connected layer with output 2 this received the following accuracy and f1_score:
67% accuracy f1-0.38 This was the one I used for fine tuning.

Xception fine tuning

I froze the bottom 116 layers of the Xception model, and used my fully connected layer that I trainined in the last section, this obtained an Accuracy of 63.33% Fl score of 0.41.

ResNet50 intial model training

After initial training varying several parameters, and trialing several models I managed to obtain an accuracy of 80% and an f1-score of 0.46.

ResNet50 fine tuning

When only the 5th stage of the ResNet50 model was unfrozen after 100 epochs no improvement was made above the initial model training, so I decided to attempt to unfreeze the 4th stage as well.
After 50 epochs of training it looked as if the model was beginning to overfit to the training set and no improvement in f1 score was being seen against the validation set so I decided to terminate.
I then tried to freeze half the 4th layers weights, and a smaller batch size of 20 to see if that made any improvement.

The best model when applied to the testing data set showed no improvement to f1-score that was found during initial model training, so I decided to use the initial ResNet50 trained model for my final classifier.

InceptionV3 initial model training

I was unable to get the model to learn patterns in the data using the same parameters that I had previously used for the ResNet50 classifier, the model was biasing the benign class and resulting in an accuracy of 80% on test data when using the same configuration as specified for initial Xception classifier training. I moved to using a smaller batch size of 20, and a SGD

optimiser. I also moved the dropout rate in my fully connected layer to be 0.5. This lead to an f1 score of 0.39 and accuracy of 80.5%

InceptionV3 tine tuning

After 100 epochs of training, using the implementation described in the previous section i managed to achieve an f1 score of 0.46 and accuracy of 79.5%.

Final Model Combination

I tried using various different ways to combine the results of the CNN classifiers, these are detailed below:

Random Forest/AdaBoost/Bagging

I tried varying the number of estimators hyperparameter (1, 10, 100, 200, 500, 1000, 2000) used in each of the classifiers, and also trialling whether to use the 3 feature final classifier dataset, or the 5 feature including intermediate results. Once I found performance degraded between 2 of the points listed above, I trialled some values in between to find the optimum. For example using the AdaBoost classifier with the 3 feature input dataset.

n_estimators	F1_score on test set
100	0.44
200	0.48
500	0.47
400	0.48
300	0.49
340	**0.5**

The best performing classifiers are listed in the results section below.

SVC

Since SVC's have many potential tunable parameters I decided to speed up the tuning process using GridSearchCV, which trains and validates a model using every combination of hyper parameters specified and using a proportion of the training data to validate results. The parameters and values i decided to trial were:

kernel : [rbf, linear],
gamma (for rbf kernel): 1e-3, 1e-4],
C : [1, 10, 100, 1000]

The optimum set of hyper parameters for the 3 feature and 5 feature dataset were:
{'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}

The best performing classifier performances are listed in the results section below.

IV. Results

Model Evaluation and Validation

The evaluation metrics in this section will be used to access which of the models, and intermediate models that I trained during this investigation were most suitable for being used to diagnose malignant melanomas. To ensure the results could be trusted the metrics have been calculated by using each model to predict whether an image is benign or malignant from an unseen testing dataset of 600 images.

I believe these results are robust, because I have used a hidden testing set to access the performance.

This table lists the metrics for the final ensemble/combination techniques I tried to merge the features of my CNN's that were trained by transfer learning:

Model Type, feature set 3 or 5 (optimum hyperparameters)	**Sensitivity (recall)**	**Specificity**	**F1 score**	**Precision**	**Accuracy**
Random Forrest Classifier 3 (n_estimators=500)	0.39	0.89	0.43	0.49	0.79
Random Forrest Classifier 5 (n_estimators=400)	0.40	0.90	0.44	0.51	0.80
AdaBoost Classifier 3 (n_estimators=340)	**0.44**	**0.92**	**0.50**	**0.57**	**0.83**
AdaBoost Classifier 5 (n_estimators=620)	0.38	0.92	0.44	0.52	0.81
Bagging Classifier 3 (n_estimators=600)	0.39	0.90	0.43	0.50	0.80
Bagging Classifier 5 (n_estimators=2000)	0.37	0.91	0.43	0.50	0.80
SVC 3 (kernel=rbf, c=100, gamma= 0.001)	0.42	0.89	0.45	0.51	0.80
SVC 5 (kernel=rbf, c=100, gamma= 0.001)	0.42	0.89	0.45	0.51	0.80

Judging by these metrics the ensemble method which performed the best was the Ada Boost classifier using the features of the 3 final CNN's as input. This scored highest for all of the performance metrics, the most important being the sensitivity (proportion of Malignant samples predicted correctly).

This table lists the performance metrics of the best ensemble method as well as each intermediate and fine tuned CNN.

Model Type	Sensitivity (recall)	Specificity	F1 score	Precision	Accuracy
AdaBoost Ensemble	0.44	0.92	**0.50**	**0.57**	**0.83**
InceptionV3 Intermediate	0.31	**0.93**	0.39	0.48	0.81
InceptionV3 Final	0.45	0.88	0.46	0.52	0.80
Resnet50 Final	0.44	0.89	0.46	0.51	0.80
Xception Intermediate	0.50	0.72	0.38	0.45	0.67
Xception Final	**0.65**	0.63	0.41	0.51	0.63

The best performing model for this problem set without adjusting decision boundaries would have been Xception Final, even though it had the worst accuracy because it had a much better sensitivity score of the malignant target variable on the test set.

With exception of the Xception classifiers (sensitivity/specificity scores appear similar in that case), every other classifier appears heavily biased to the benign output class, this has resulted in high specificity scores but low sensitivity scores on the test dataset. I believe this is due to the imbalance of benign/malignant classes in the training and validation data sets (4:1), even though I had attempted to use the class_weights feature in keras to weight the loss function to attempt to balance the difference. Alternate approaches to address this imbalance will be discussed in the conclusion below.

Justification

I have calculated the same performance metrics used to evaluate my models for the benchmark results presented earlier in this report, I have also presented the scores that would have been achieved if random chance was used to predict the output feature. The calculations can be found in the appendix.

Model Type	Sensitivity (recall)	Specificity	F1 score	Precision	Accuracy
AdaBoost Ensemble	0.44	**0.92**	0.50	0.57	0.83
Xception Final	0.65	0.63	0.41	0.51	0.63
Esteva et al, CNN	**0.96**	0.90	**0.93**	**0.91**	**0.93**
Clinical average	0.95	0.76	0.87	0.80	0.86
Random	0.5	0.5	0.29	0.2	0.5

The benchmark results of the CNN Esteva et al trained performs significantly better than any model that I was able to train, this was to be expected, since the team had access to significantly more training data and computing resources and they were highly experienced data scientists. The Esteva et al CNN has a sensitivity of 0.96, specificity of 0.90 and an overall accuracy of 0.93, which is very impressive and outperforms the clinical average they used as a benchmark.

This clinical average also performs significantly better than my best classifiers when you compare the sensitivity values of being able to identify the target class (malignant) correctly, which of course is the most important thing for a clinical test (a false positive test outcome, someone has a biopsy/lesion removed that is not actually cancerous, has a significantly lower cost than a false negative outcome, a malignant lesion is missed and so not treated leading to a patient's condition worsening/death).

Using the Roc Curve visualisation that I have generated, and is included in the conclusion below, this showed that by shifting the decision boundary to get a sensitivity rate comparable to the clinical average 0.95, would result in a specificity of 0.28 which would be far too low to use it reliably as a clinical test. Whilst 95% of people with malignant melanomas would be correctly treated it would result in 72% of people who had a benign lesion being sent for a biopsy when they didn’t need one adding significant cost to the NHS and causing undue stress and pain to an unproportionate number of people.

V. Conclusion

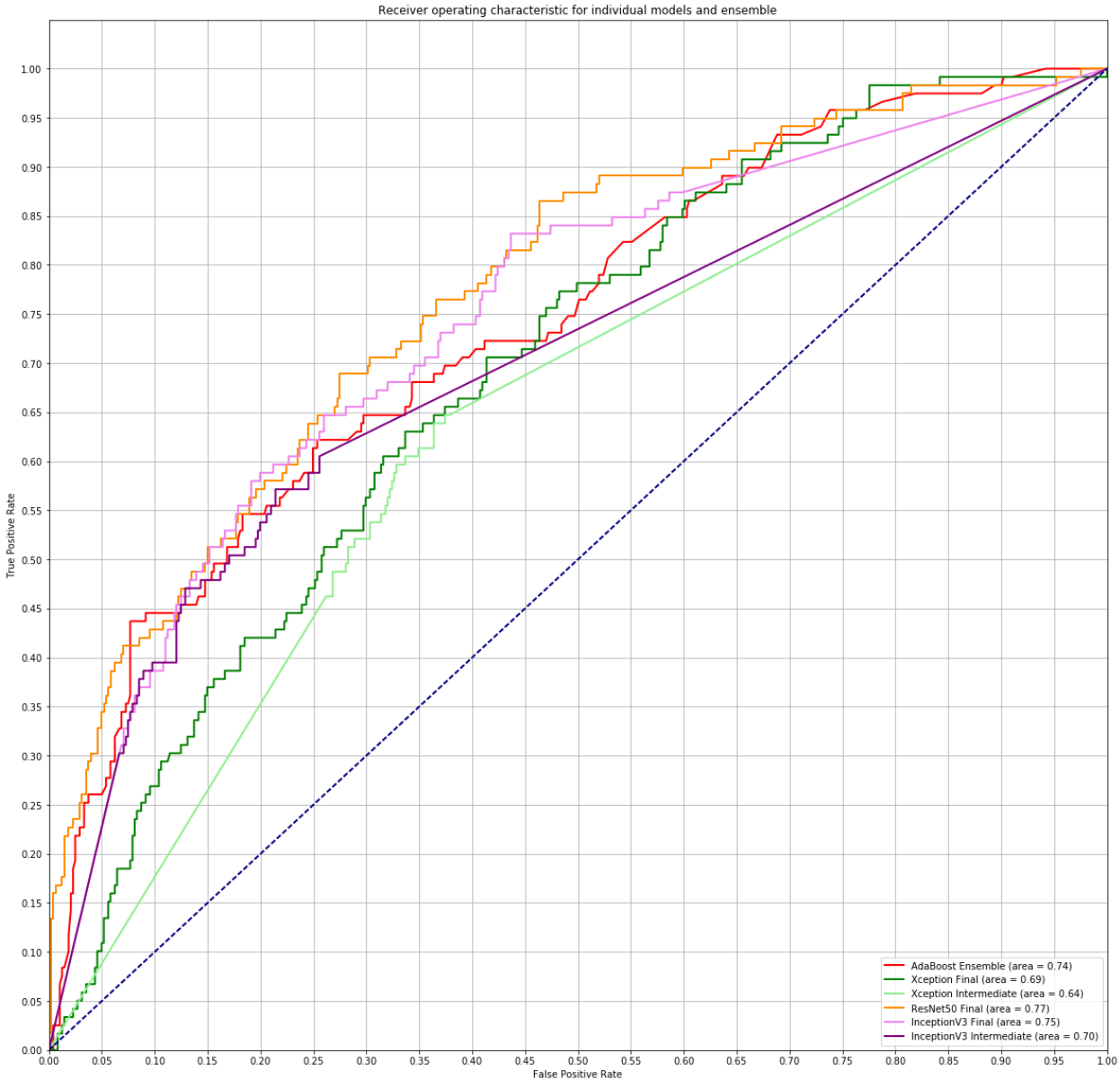
Free-Form Visualization

Roc curves are a good way to access the performance of binary classifiers by plotting true positive rate (sensitivity/recall) against the false positive rate (1- specificity) for different decision thresholds of predictions made by a classifier. They are useful for deciding on the decision

boundary that is most suitable for the business problem that is trying to be addressed, helping to decide on an acceptable tradeoff between sensitivity and specificity.

Better performing models occur further towards the top left hand part of the plot, closest to the point (0.0, 1.0). The dashed line that runs through the origin and (0.5,0.5) represents random chance in predicting an output class. A perfect classifier (where TRP = 1.0 and FPR = 0.0) would result in a horizontal line that goes though (0.0, 1.0).

Below you can see each of the ROC curves from the CNN models created and also the best performing ensemble method.



Firstly all of the models I have created perform better than the random chance distribution denoted by the dashed line. If you look at when the true positive rate (TPR) is 0.5 (50% of malignant samples predicted correctly), the random chance line has a false positive rate (FPR) of 0.5 (50% of the benign samples are predicted correctly) whereas my worse performing classifier (Xception Intermediate) has a FPR around 0.27 (73% of benign samples are predicted correctly), and my best performing classifiers (ResNet50 final/InceptionV3 final) has a FPR 0.15 (85% of benign samples are predicted correctly).

I have calculated the AUC (area under the curve) score for each model that can be used to compare the overall performance of a classifier, this shows the proportion of the space that appears beneath the ROC curve. ROC curves that have a higher AUC score are generally considered better classifiers. The random chance classifier would obtain an AUC score of 0.5. The highest performing classifier using this as a measure would be ResNet50 final, which has an AUC of 0.77, followed by InceptionV3 final with 0.75, and the AdaBoost Ensemble with 0.74.

Whilst this is a useful statistic to provide a general comparison, since this project is about providing diagnosis for a medical test the best classifier would obtain a high sensitivity score and comparably high specificity score. If we chose a TPR that may be suitable for a medical test, say 0.95 which is similar to that of clinicians [15], the random classifier has a FPR 0.95 (only 5% of benign samples are predicted correctly), and my best performing classifier at this rate (ResNet50 final) has a FPR 0.72 (28% of benign samples are predicted correctly). This FPR is far too high for the classifier to be considered usable in a real world medical context, but I think it is probably the best performing classifier I have trained for this project. The ensemble method I used to combine results of all of the prior CNN's did not perform better than the best performing CNN for this problem.

Reflection

My solution to this problem was to train convolutional neural networks using transfer learning and then combine the results using an ensemble method. I trained 3 separate initial models based on the Xception classifier, ResNet50 classifier and InceptionV3. I then attempted to fine tune the performance of these models by freezing lower network layers and retraining based on the weights I had learnt when training the initial models. I then formatted the output predictions so that I could use them as input for an ensemble method. I tried several and the most performant was AdaBoost. The performance of the best performing CNN (ResNet50) was slightly better than the AdaBoost classifier when using a ROC curve and comparing based on sensitivity near an acceptable threshold of a medical practitioner (0.95).

The most challenging aspect was the time required to train the CNN models. I used my own laptop to do this, if I had more money to spend I would have used AWS with GPU based

machines optimised for machine learning with tensorflow. The would enable me to try out different CNN architectures and training parameters more quickly.

Another challenging aspect was trying to deal with unbalanced data in my training datasets. Initially when I tried training my CNN's they would quickly narrow on a solution which predicted all images as the class with the highest proportion of samples in the training set (benign 80%) achieving an accuracy of 80%. To try and get round this I used the class_weights parameter in keras during training to try and address the imbalance by weighting the loss function based on the ratio of classes. In my final results it still appears that the classifiers are still biased towards the benign class so I think more improvements could be made to address this.

I think the concept of using an ensemble of CNN's for this type of problem is still sound, and CNN's are still used widely for image classification problems. If I were able to train less biased classifiers I would have obtained results closer to the clinician benchmark stated in the project. I think I would still struggle to get anywhere near the performance of the CNN trained by Esteva et al [15].

Improvement

There are several other things that I could have tried to attempt to overcome the problem with unbalanced training data. From research it appears that micro batches in Keras do not contain an even proportion of examples from each class [18], so each micro batch would have the same distribution of examples as the overall training set, this may have led to bias during the training process. To overcome this I could have either reduced the amount of benign training examples so that there were an equal proportion of both or created additional training data. My prefered method would be the later creating additional training and validation data from the original malignant images, providing transformations such as rotation and scaling, so that I had an even proportion of images of each class. Then in each micro batch the proportion of each class would be even.

Additional training data would also be favourable to help the CNN's learn patterns from the data. In general it is widely considered that the more data there is the better, the limiting factor would be time to train and cost. I could have downloaded more examples from the ISIC archive, or produced more augmentations using the method described above. To handle the increased data volumes and the longer resulting training times I would use dedicated AWS machine learning instances with GPU's to train the models, but would have to be mindful of cost.

References

[1] http://www.who.int/uv/faq/skincancer/en/index1.html

[2] http://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/skin-cancer#heading-Zero

[3] https://www.nhs.uk/be-clear-on-cancer/symptoms/skin-cancer#QxQ74ksgVC7py2mo.97

[4] https://en.wikipedia.org/wiki/Sensitivity_and_specificity#Sensitivity

[5] https://en.wikipedia.org/wiki/Receiver_operating_characteristic

[6] https://uberpython.wordpress.com/2012/01/01/precision-recall-sensitivity-and-specificity/

[7] http://isdis.net/isic-project/

[8] https://isic-archive.com/#images

[9] http://girder.readthedocs.io/en/latest/user-guide.html

[10] https://challenge.kitware.com/#phase/5840f53ccad3a51cc66c8dab

[11] https://en.wikipedia.org/wiki/Convolutional_neural_network

[12] https://arxiv.org/abs/1610.02357

[13] https://arxiv.org/abs/1512.03385

[14] https://arxiv.org/abs/1512.00567

[15] https://www.nature.com/articles/nature21056

[16] https://keras.io/

[17] https://www.tensorflow.org/

[18] http://scikit-learn.org/

[14] https://keras.io/applications/#available-models

[15] http://scikit-learn.org/stable/modules/ensemble.html#forest

[16] https://towardsdatascience.com/fine-tuning-a-classifier-in-scikit-learn-66e048c21e65

[17] https://keras.io/applications/#fine-tune-inceptionv3-on-a-new-set-of-classes

[18] https://github.com/rstudio/keras/issues/366

Apendix

Calculations for benchmark performance metrics

Here are the calculations to generate f1_score, accuracy and precision for these benchmark values so that I can compare with my results.

Esteva et al, CNN (malignant 96%, benign 90%)
Confusion matrix:

		Actual Class	
		Malignant	Benign
Predicted Class	Malignant	(TP) **96**	(FP) **10**
	Benign	(FN) **4**	(TN) **90**

Sensitivity - 0.96
Specificity - 0.90
Precision - TP/TP+FP = 96/(106) = 0.91
Accuracy - TP + TN / TP + TN + FN + FP = 0.93
F1_score - 2 * (Precision * recall)/(Precision + recall) = 0.93

Clinician (malignant 95% accuracy, benign 76% accuracy)
Confusion matrix:

		Actual Class	
		Malignant	Benign
Predicted Class	Malignant	(TP) **95**	(FP) **24**
	Benign	(FN) **5**	(TN) **76**

Sensitivity - 0.95
Specificity - 0.76
Precision -TP/TP+FP = 95/(119) = 0.80
Accuracy - TP + TN / TP + TN + FN + FP = 0.86

F1_score - 2 * (Precision * recall)/(Precision + recall) = 0.87

Random
50% chance of selecting the right class
Given 80 Benign and 20 malignant.

		Actual Class	
		Malignant	Benign
Predicted Class	Malignant	(TP) **10**	(FP) **40**
	Benign	(FN) **10**	(TN) **40**

Sensitivity - 0.5
Specificity - 0.5
Precision - 10/10+40 = 0.2
Accuracy - 10+40/100 = 0.5
F1 2 * (0.2 * 0.5)/ (0.2 + 0.5) = 0.29