Machine Learning Engineer Nanodegree

Skin Cancer Detection Capstone Report

Srinivasa Amirapu
Dec 9th, 2018

Definitions

Project Overview

Skin cancer is the most common cancer. Each year there are more new cases of skin cancer than the combined incidence of cancers of the breast, prostate, lung and colon. Over the past three decades, more people have had skin cancer than all other cancers combined. Skin cancer is a major public health problem, with over 5 million newly diagnosed cases in the United States each year. Melanoma is the deadliest form of skin cancer, responsible for over 9,000 deaths each year. The estimated 5-year survival rate for patients whose melanoma is detected early is about 98 percent in the U.S. The survival rate falls to 62 percent when the disease reaches the lymph nodes, and 18 percent when the disease metastasizes to distant organs.

As the computers are getting better at understanding the images due to advances in Deep Learning , We can apply Deep Learning with Transfer Learning to develop models which help doctors in early detection of skin cancer. To err is human, to forgive is divine. But to forgive an human error we need a better backup from a AI machine to validate our errors . In the current context we do not want to send sick patients home i.e., patients with malignant tumor should not be misclassified and sent home.

In this Project, I created predictive model for skin cancer detection which can help in increasing survival rate of the patients. This predictive model for skin cancer detection can help doctors to better serve patients by offering them pre-screening.

Problem Statement

The goal of the challenge is to develop image analysis tools to enable the automated diagnosis of melanoma from dermoscopic images. The main objective is to design an algorithm that can visually diagnose melanoma, the deadliest form of skin cancer. Our algorithm will distinguish this malignant skin tumor from two types of benign lesions (nevi and seborrheic keratoses). The data and objective are pulled from the 2017 ISIC Challenge on Skin Lesion Analysis Towards Melanoma Detection. As part of the challenge, participants were tasked to design an algorithm to diagnose skin lesion images as one of three different skin diseases (melanoma, nevus, or seborrheic keratosis). In this project, we will create a model to generate our own predictions.

Datasets and Input

The data is pulled from the 2017 ISIC Challenge on Skin Lesion Analysis Towards Melanoma Detection of International Skin Imaging Collaboration.

Initial Datasets were downloaded from the Udacity repository by following below steps:-

1. Clone the repository and create a data/ folder to hold the dataset of skin images.

2. git clone https://github.com/udacity/dermatologist-ai.git

3. mkdir data; cd data

4. Create folders to hold the training, validation, and test images.

5. mkdir train; mkdir valid; mkdir test

6. Download and unzip the training data (5.3 GB).

7. Download and unzip the validation data (824.5 MB).

8. Download and unzip the test data (5.1 GB).

9. Place the training, validation, and test images in the data/ folder, at data/train/, data/valid/, and data/test/, respectively. Each folder should contain three sub-folders (melanoma/, nevus/, seborrheic_keratosis/), each containing representative images from one of the three image classes.

Data augmentation

We used codebox python utility (https://codebox.net/pages/image-augmentation-with-python) to scan directory containing image files, to generate images by performing a specified set of augmentation operations on each file that it finds. This process multiplies the number of training examples that can be used to significantly improve the resulting network's performance, particularly when the number of training examples is relatively small. We boosted image set with total skin images of 14307. Out of which training set included skin cancer images of 13557.There are 150 validation skin cancer images and 600 test skin cancer images.

Below Data augmentation operations were performed, using the codes listed in the table below:

Code	**Description**	**Used Values**
fliph	Horizontal Flip	fliph
flipv	Vertical Flip	flipv
noise	Adds random noise to the image	noise_0.01 noise_0.5
rot	Rotates the image by the specified amount	rot_90 rot_-45
trans	Shifts the pixels of the image by the specified amounts in the x and y directions	trans_20_10 trans_-10_0
zoom	Zooms into the specified region of the image, performing stretching/shrinking as necessary	zoom_0_0_20_20 zoom_-10_-20_10_10
blur	Blurs the image by the specified amount	blur_1.5

Preprocessing:

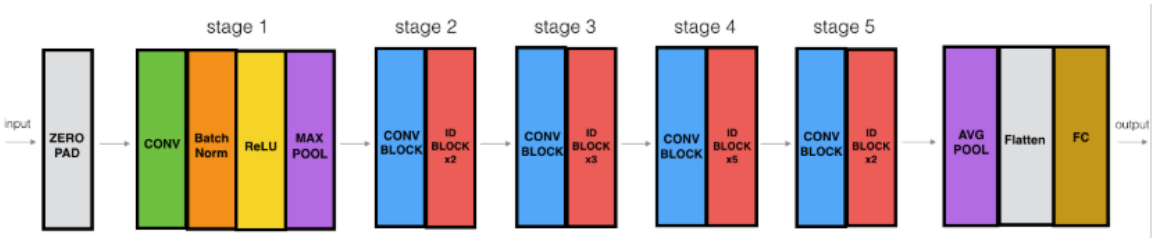
When using TensorFlow as backend, Keras CNNs require a 4D tensor as input, with shape (nb_samples, rows, columns, channels) where nb_samples corresponds to the total number of images (or samples), and rows, columns, and channels correspond to the number of rows, columns, and channels for each image, respectively.
The path_to_tensor function in the notebook takes a string-valued file path to a color image as input and returns a 4D tensor suitable for supplying to a Keras CNN.
The function first loads the image and resizes it to a square image that is 224*224 pixels. Next, the image is converted to an array, which is then resized to a 4D tensor. In this case, since we are working with color images, each image has three channels. Likewise, since we are processing a single image (or sample), the returned tensor will always have shape(1,224,224,3).
The paths_to_tensor function takes a numpy array of string-valued image paths as input and returns a 4D tensor with shape (nb_samples,224,224,3).
Here, nb_samples is the number of samples, or number of images, in the supplied array of image paths.

Implementation:

Model Architecture:ResNet Model with 50 layers

To model image classifier we would need very deep network like the one used in the referenced publication -Dermatologist-level classification of skin cancer with deep neural networks* which used InceptionV3 model to represent very complex functions. Very deep "plain" networks don't work in practice because they are hard to train due to vanishing gradients.

Residual Networks, introduced by He et al., allow you to train much deeper networks than were previously practically feasible.The skip-connections help to address the Vanishing Gradient problem. They also make it easy for a ResNet block to learn an identity function.There are two main type of blocks: The identity block and the convolutional block.Very deep Residual Networks are built by stacking these blocks together.



Create a CNN to Classify Skin Cancer (using Transfer Learning)

We used a proven technique - transfer learning to create a CNN that can classify images into melanomas, nevus, or SBK. Our CNN attained more than 70% accuracy on the test set.

We applied transfer learning to create a CNN Model using bottleneck features obtained from ResNet-50 pre-trained network trained on ImageNet data (unlike in the reference publication -Dermatologist-level classification of skin cancer with deep neural networks* which used Inception-V3 model) :

We further performed below steps to fine tune our CNN model to yield better accuracy:

1. Image augmentation: Boosted image dataset by implementing Rotations, noising, scaling
2. Transfer Learning: We added a global spatial average pooling layer to the last convolutional output of ResNet50 Model trained on ImageNet followed by fully-connected layer with 'relu' activation and a fully connected layer, where the latter contains one node for each skin cancer category(melanomas, nevus, or SBK) and is equipped with a softmax.

Evaluation Metrics

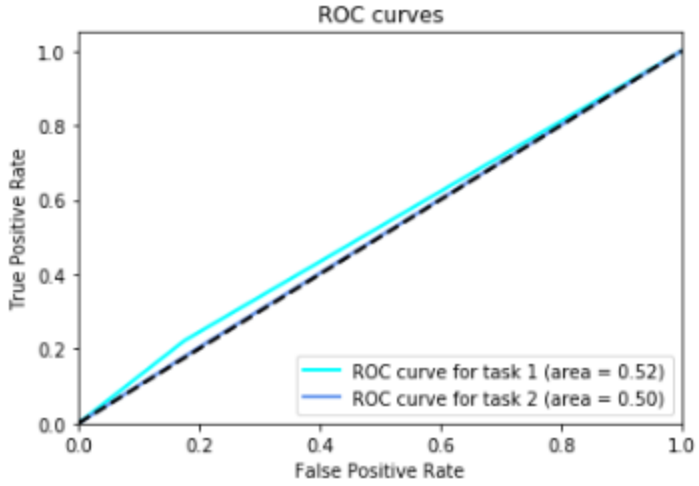
ROC AUC for Melanoma Classification

To evaluate the different models, I will use ROC Curves and AUC score. To choose the correct model I will evaluate the precision and accuracy to set the threshold level that represent a good tradeoff between TPR and FPR. We will gauge the ability of your CNN to distinguish between malignant melanoma and the benign skin lesions (nevus, seborrheic keratosis) by calculating the area under the receiver operating characteristic curve (ROC AUC) corresponding to this binary classification task.

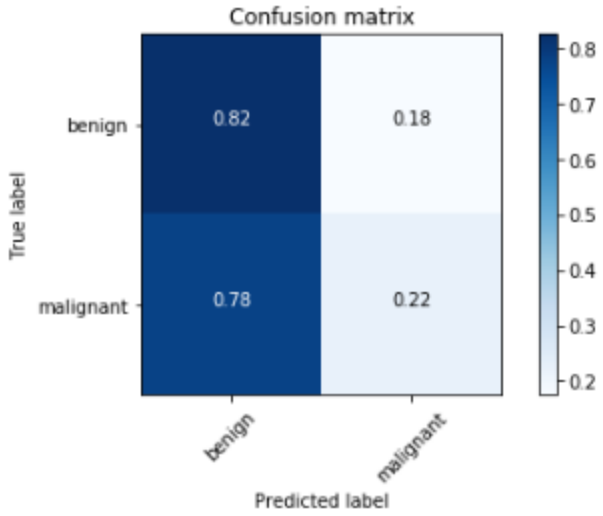
Results

Classification Report ResNet50 with Data Augmentation.

- Model_name = ResNet50
- 100 epochs.ModelCheckpoint. Best Val Accuracy
- AUC: 0.52




```
Category 1 Score: 0.523
Category 2 Score: 0.502
Category 3 Score: 0.513
```



Reflection

My reason for choosing ResNet50 architecture is that ResNet50 model deals with vanishing gradients problem better than other models and also reduces overfitting problem along with number of parameters needed for computation. But in this experiment it resulted in poor AUC score but I believe the score can further be improved by adding batch normalization and shuffling the train set after data augmentation and also adding more images to reduce overall imbalance and overfitting.

The most difficult part for me was to get the experiments running on AWS EC2 Instance with GPU mode enabled. Although computational time was reduced with GPU option, there is cost involved which resulted in lower number of experiments being done.

5.3. Improvement

We can further improve the accuracy by tuning certain hyperparameters and choosing different optimizations techniques.

Due to time and computational cost it was not possible for me to run more experiments using different known architectures other than Resnet50 for this dataset. It's definitely possible that a different architecture would be more effective. Given enough time and computational power, I'd definitely like to explore the different approaches.

As the classes were heavily imbalanced, one of my hypotheses is if I generate further augmented dataset for the classes that have less data than the others, save them in the training set would help to improve model further.

References :

- https://codebox.net/pages/image-augmentation-with-python
- https://challenge.kitware.com/#challenge/583f126bcad3a51cc66c8d9a
- https://www.udacity.com (Dermatologist AI)
- Dermatologist-level classification of skin cancer with deep neural networks by Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau & Sebastian Thrun
- Coursera DeepLearning.ai course