# Machine Learning Engineer Nanodegree

## Capstone Project

Miodrag Vujković
June 5th, 2017

# I. Definition

## Project Overview

Melanoma is the most dangerous type of skin cancer. Wikipedia definition is the following: "**Melanoma**, also known as **malignant melanoma**, is a type of cancer that develops from the pigment-containing cells known as melanocytes. Melanomas typically occur in the skin but may rarely occur in the mouth, intestines, or eye. In women they most commonly occur on the legs, while in men they are most common on the back. Sometimes they develop from a mole with concerning changes including an increase in size, irregular edges, change in color, itchiness, or skin breakdown."

This project focuses on differentiating benign from malignant skin lesions using an android app. App is based on a tensorflow classifier trained on ISIC-archive (https://isic-archive.com/) gallery of skin lesion images. Classifier is an Inception v3 model retrained on mentioned images.

Inspiration for this project came from work published by Stanford Artificial Intelligence Laboratory researchers (Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau & Sebastian Thrun) in Nature magazine (Dermatologist-level classification of skin cancer with deep neural networks).

The app is based on Tensorflow mobile demo app provided in the tensorflow github repo (tensorflow on android) and Docker image provided by Dan Jarvis (blog post).

## Problem Statement

Project goal is to make an android app that will classify skin lesion as benign of malignant based on the phone camera input. Following steps were performed:

- Downloaded image datasets from ISIC-archive
- Prepared and augmented images for training
- Trained the final layer of Inception V3 network
- Stripped the model in order to use it in the android app
- Built and tested the mobile app

Downloading the dataset and images metadata was time consuming since this is a 40GB dataset with more than 13,000 images.

Training the classifier on new images was not that much time consuming since I used a pretrained Inception v3 model from GoogLeNet and only retrained the last layer.

## Metrics

Test accuracy is used for metrics. Accuracy is the simple but effective measure of model success. Since I only have two data classes, my opinion is that it is sufficient to assess usability of the underlying model.

It is computed as (true positives + true negatives) / number of test cases.

Model achieves 88.5% accuracy after 100,000 training steps.

# II. Analysis

## Data Exploration

Dataset consists of 13,786 images divided into 9 subdatasets/folders:

- ISIC_UDA-1_1 (557 images of moles and melanomas. Biopsy confirmed. Contains both malignant and benign lesions)
- ISIC_UDA-2_1 (60 images of moles and melanomas. Biopsy confirmed. Contains both malignant and benign lesions)
- ISIC_SONIC_1 (9251 images of benign melanocytic moles in pediatric population)
- ISIC_MSK-1_1 (683 images of moles and melanomas. Biopsy confirmed. Contains both malignant and benign lesions)
- ISIC_MSK-1_2 (417 images of malignant and bening melanocytic and non-melanocytic lesions confirmed by histopathology)
- ISIC_MSK-2_1 (1535 images of melanocytic and non-melanocytic lesions. Biopsy confirmed)
- ISIC_MSK-3_1 (225 images of mostly nevi and basal cell carcinomas confirmed by histopathology)
- ISIC_MSK-4_1 (947 images of lesions confirmed by histopathology)
- ISIC_MSK-5_1 (111 images of seborrheic keratosis determined by concensus of three experts)

Dataset contains the following structure of bening and malignant images:

| bening | malignant | indeterminate |
|---|---|---|
| 12,670 | 1,082 | 34 |

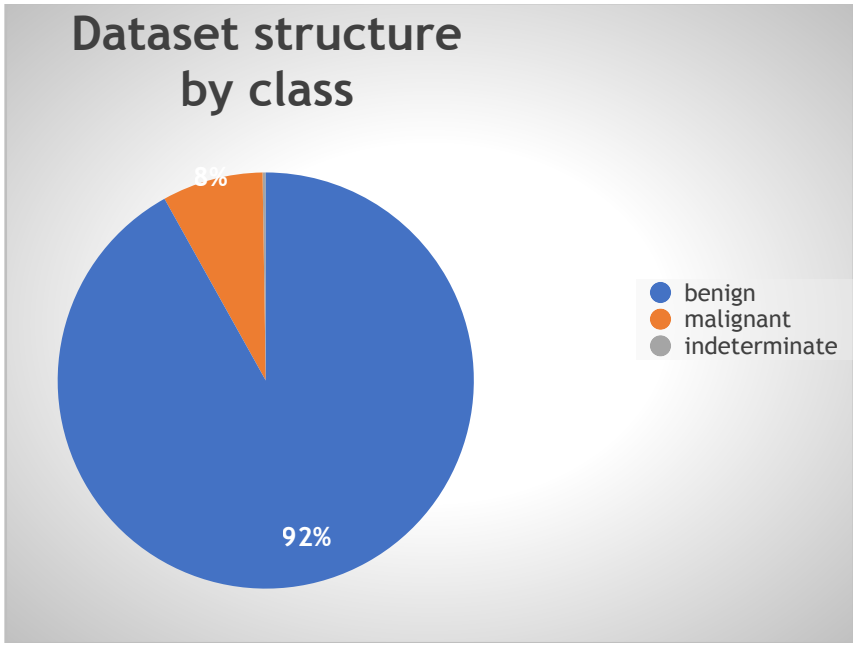
We can see that dataset is highly unbalanced with around 92% of images being bening and 8% being malignant. Indeterminate images were discarded during the training.

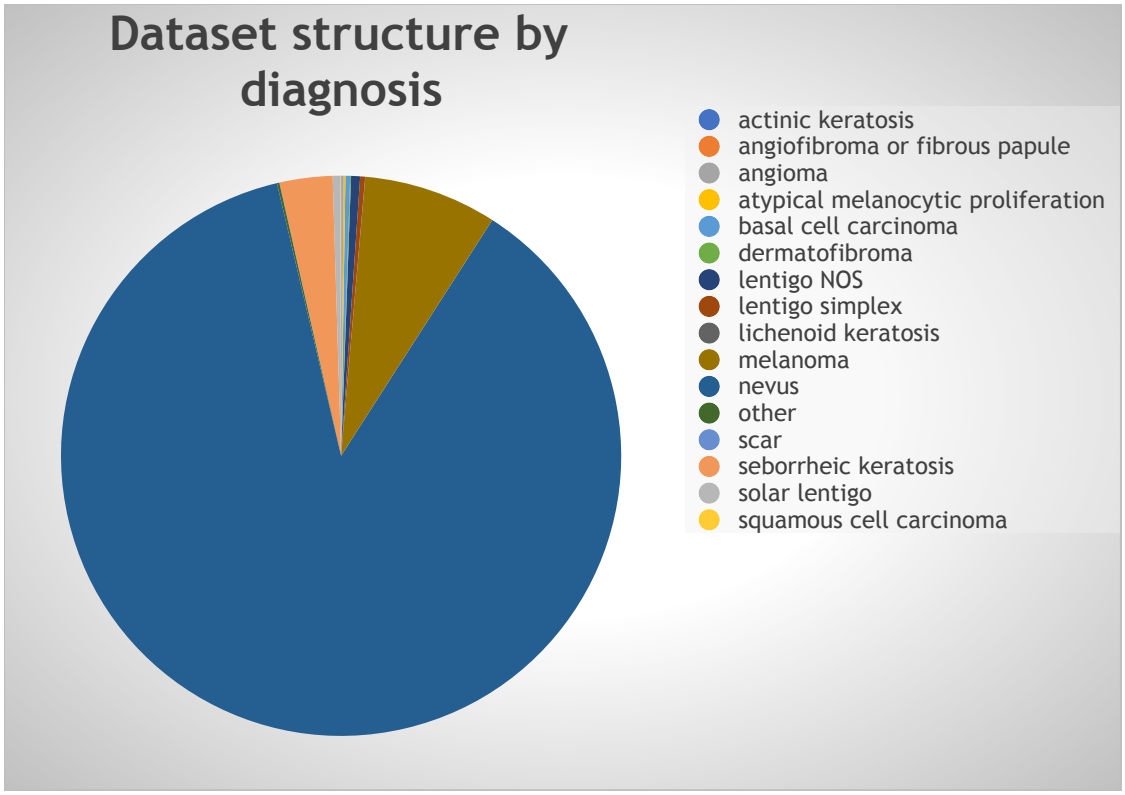
Dataset contains the following structure of images by diagnosis:

| diagnosis | count |
|---|---|
| actinic keratosis | 2 |
| angiofibroma or fibrous papule | 1 |
| angioma | 17 |
| atypical melanocytic proliferation | 14 |
| basal cell carcinoma | 37 |
| dermatofibroma | 7 |
| lentigo NOS | 71 |
| lentigo simplex | 39 |
| lichenoid keratosis | 1 |
| melanoma | 1056 |
| nevus | 11965 |
| other | 19 |
| scar | 1 |
| seborrheic keratosis | 419 |
| solar lentigo | 61 |
| squamous cell carcinoma | 5 |

As we can see the dataset is highly unbalanced regarding specific diagnosis with majority of images belonging to three classes (melanoma, nevus and seborrheic keratosis). Some diagnosis only have one image presented. It is highly unlikely that neural network would learn to classify images correctly by diagnosis by training solely on this dataset. Therefore, I decided to use only two classes: benign and malignant.

## Exploratory Visualization



## Algorithms and Techniques

The classifier that I used is a pre-trained Inception V3 from GoogLeNet. It was trained on ImageNet data from 2012 (around 1.4 million images). These data contain 1000 classes. It reached 3.46% of top-5 error rate on the mentioned dataset which is beyond human accuracy (measured by Andrej Karpathy (blog post).

I used "transfer learning" by removing the last layer of the network and retraining it on my image dataset. For this, I used the script provided in the tensorflow repository. Repository and script are contained within Docker container provided by Dan Jarvis (blog post).

Final model was stripped of unused operations in order to be usable on mobile devices.

Model graph (.pb) and labels(.txt) files were then copied to Android Studio project based on the example provided in Tensorflow Github repository.

I changed an example app in regard to the following:

- Removed unnecessary activities that were used for object tracking and style transfer,
- Changed some colors in the UI and launcher icon,
- Used models trained on skin lesions images instead of ones provided with the app (by not using download-models.gradle file),
- Adjusted the classifier to work with V3 of the Inception model (example app works with Inception V1, so I had to change image size, mean and std)

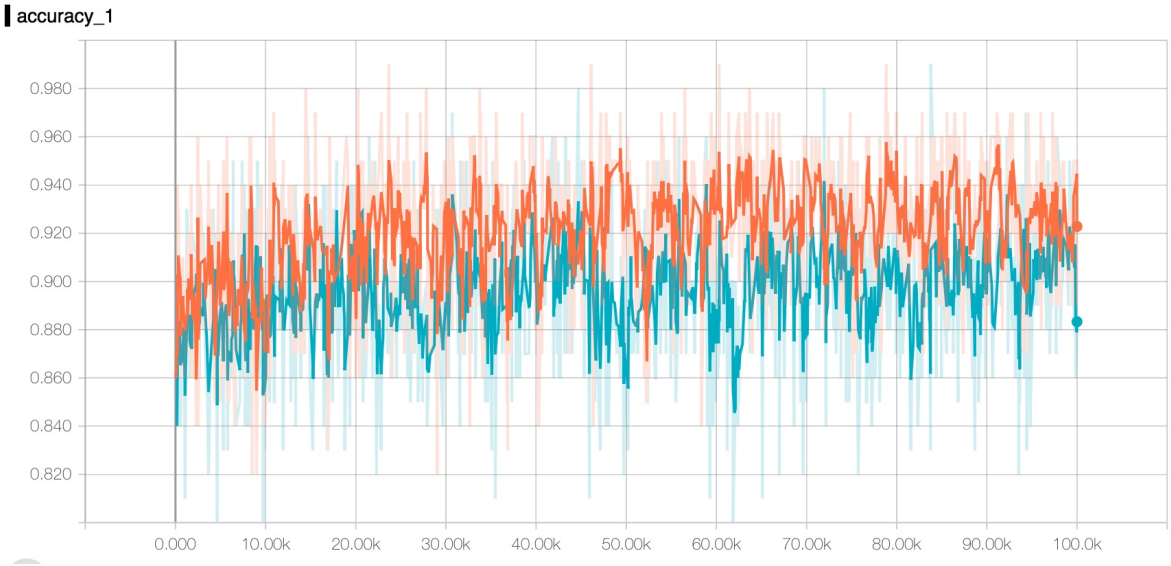
## Benchmark

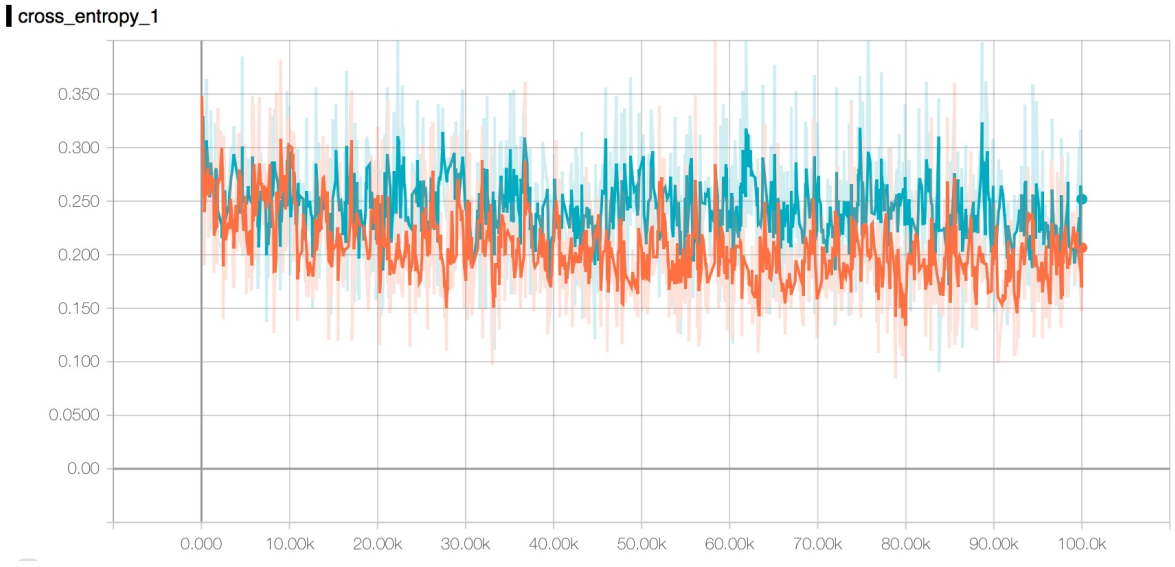
I could not find the study or project that used the same approach as me. Stanford Artificial Intelligence Laboratory used a more complicated approach with more data (129,450 images) and classified images to 2,032 diseases.

They validated the algorithm in several steps, first using the three classes partitions (benign, malignant and non-neoplastic achieving 72.1 ± 0.9% of accuracy surpassing a results of two dermatologists). Second level of validation was done on nine partition classification achieving 55.4 ± 1.7% also surpassing the two dermatologists.

My algorithm achieved 88.5% test accuracy for classification on just two classes (benign and malignant). Augmented dataset was split into training (80%), validation (10%) and test (10%) datasets.

Accuracy and cross entropy (tensorboard):



# III. Methodology

## Data Preprocessing

Original images are downloaded from ISIC-archive. These are mostly high resolution photographic images of skin lesions.

In order to be used in Inception v3 model, images had to be resized to 299x299 pixels. Also, to get more training data, I use some image altering techniques to augment the dataset.

Every image was:

- Resized to 299 x 299 pixels,
- Flipped horizontally,
- Flipped vertically,
- Rotated 90 degrees,
- Rotated 180 degrees,
- Rotated 270 degrees, and

original image and 5 variations were saved to the augmented directory.

I also tried rotating images in different angles but it decreased the accuracy of the model. Also, I did not use geometric transformations, since symmetry and shape of the lesion edges are very important in classifying the lesion as benign or malignant. Classical method for determining melanoma is called ABCDE method. It is an acronym for:

- A – asymmetrical shape,
- B – border (irregular borders),
- C – color (more than one color or uneven distribution),
- D – diameter (greater than 6 mm in diameter), and
- E – evolution (changes in color and size).

## Implementation

First part of implementation included downloading images, metadata, preprocessing and augmenting the dataset.

It took a long time since this is a 40 GB dataset with almost 14,000 images.

After that, neural network (Inception V3 Convolutional Neural Network) was trained on augmented dataset. The most time consuming step in this phase was creating bottlenecks for images. This step was done only once and subsequent experimentations with hyper-parameters were much faster that the first training.

Resulting model was stripped and used as a classifier in android app.

Android app is an altered version of example app provided in the tensorflow github repository.

## Refinement

I tried different techniques for augmenting the dataset, but most of them did not work very well, so I ended up with basic flipping and rotating.

Since this is a pre-trained network, it reached solid results very fast (after several hundred training steps) and improved only slightly afterwards.

I settled for 100,000 training steps which gave me 88.5% of test accuracy.  In example, 20,000 training steps gave me 87.2% accuracy.

Also, I tried varying some hyper-parameters but it did not work, so i came back to default ones:

- Learning rate 0.01

- Train batch size 100

# IV. Results

## Model Evaluation and Validation

Dataset was split into 80% training, 10% validation and 10% testing split. Final test accuracy was 88.5%.

Training and validation accuracy are presented in tensorboard graphs above.

## Justification

Having in mind that model is trained on a relatively small dataset of visually very similar skin lesions, it is achieving pretty good accuracy.

A mobile app with this kind of accuracy could be a pretty good preliminary diagnostic tool for detecting malignant skin lesions.

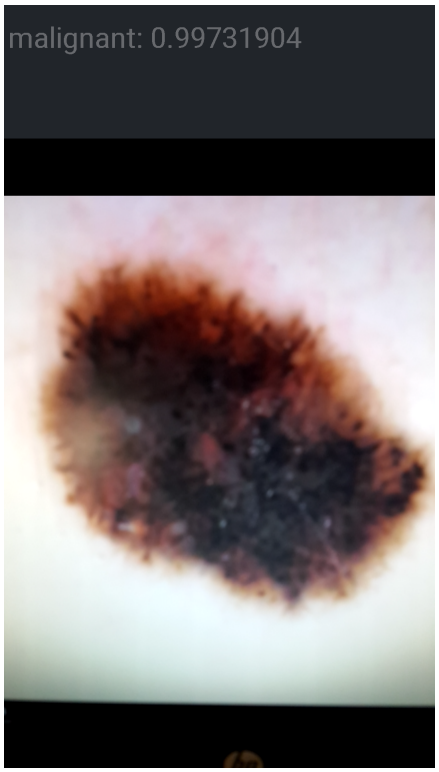
# V. Conclusion

## Free-Form Visualization

Accuracy:

```
2017-06-02 17:25:02.166522: Step 99910: Train accuracy = 91.0%
2017-06-02 17:25:02.166565: Step 99910: Cross entropy = 0.205948
2017-06-02 17:25:02.199747: Step 99910: Validation accuracy = 92.0% (N=100)
2017-06-02 17:25:02.537851: Step 99920: Train accuracy = 92.0%
2017-06-02 17:25:02.537895: Step 99920: Cross entropy = 0.200373
2017-06-02 17:25:02.572528: Step 99920: Validation accuracy = 90.0% (N=100)
2017-06-02 17:25:02.913847: Step 99930: Train accuracy = 95.0%
2017-06-02 17:25:02.913892: Step 99930: Cross entropy = 0.155597
2017-06-02 17:25:02.948905: Step 99930: Validation accuracy = 92.0% (N=100)
2017-06-02 17:25:03.289054: Step 99940: Train accuracy = 93.0%
2017-06-02 17:25:03.289097: Step 99940: Cross entropy = 0.204566
2017-06-02 17:25:03.322317: Step 99940: Validation accuracy = 91.0% (N=100)
2017-06-02 17:25:03.659911: Step 99950: Train accuracy = 95.0%
2017-06-02 17:25:03.659955: Step 99950: Cross entropy = 0.142132
2017-06-02 17:25:03.692930: Step 99950: Validation accuracy = 95.0% (N=100)
2017-06-02 17:25:04.031531: Step 99960: Train accuracy = 88.0%
2017-06-02 17:25:04.031576: Step 99960: Cross entropy = 0.259030
2017-06-02 17:25:04.064841: Step 99960: Validation accuracy = 87.0% (N=100)
2017-06-02 17:25:04.403161: Step 99970: Train accuracy = 93.0%
2017-06-02 17:25:04.403205: Step 99970: Cross entropy = 0.187692
2017-06-02 17:25:04.436270: Step 99970: Validation accuracy = 89.0% (N=100)
2017-06-02 17:25:04.777670: Step 99980: Train accuracy = 93.0%
2017-06-02 17:25:04.777713: Step 99980: Cross entropy = 0.169155
2017-06-02 17:25:04.812539: Step 99980: Validation accuracy = 91.0% (N=100)
2017-06-02 17:25:05.151637: Step 99990: Train accuracy = 92.0%
2017-06-02 17:25:05.151682: Step 99990: Cross entropy = 0.216699
2017-06-02 17:25:05.184665: Step 99990: Validation accuracy = 90.0% (N=100)
2017-06-02 17:25:05.492118: Step 99999: Train accuracy = 90.0%
2017-06-02 17:25:05.492163: Step 99999: Cross entropy = 0.257565
2017-06-02 17:25:05.526715: Step 99999: Validation accuracy = 89.0% (N=100)
Final test accuracy = 88.5% (N=8189)
Converted 2 variables to const ops.
root@2e1a14ed2fe2:/tensorflow#
root@2e1a14ed2fe2:/tensorflow#
```

Screenshot:



## Reflection

This project shows that it is not that hard to implement state of the art advancements in deep learning to real world problems.

I used:

- a publicly available dataset,
- open source pre-trained state of the art neural network (did not need to use large expensive hardware architectures for training),
- open source android app template (did not need to write an application from scratch, just changed already existing code), and

achieved pretty good accuracy in a very usable package (mobile app).

## Improvement

This project could be improved in several ways:

- accuracy: accuracy could be improved by using a bigger dataset or better augmentation strategy and maybe even a more advanced pre-trained network (such as Inception-ResNet-v2),
- .apk size: currently SkinDeep.apk is 100.2 MB large which is huge for a mobile app. Another approach could be a light client mobile app connected to web api (with tensorflow model doing the classification). Client app would take a picture, resize it and send it to REST web API or even tensorflow serving via gRPC. Since images would be 299x299 pixels, network traffic would not be that high. This API could also have a web frontend that would accept images from different camera sources. I already started working on a demo API and apps and will add them to repository when finished.