Multi-Label Skin Lesion Classification

Marina Vilela Bento (ID. 000803908) Kirill Semenov (ID. 341073617)

Submitted as final project report for the Deep Learning course, IDC, 2020

1 Introduction

Skin cancer is, by far, the most common cancer in the world and the accuracy of its diagnosis remains a challenge. Even though this form of cancer is highly treatable when found early, dermatologists rarely achieve sensitivity exceeding 80% in skin cancer screening settings [7]. Deep Learning techniques have an immense potential to assist with the early identification of this disease. For our project we decided to test such potential ourselves.

1.1 Related Works

Convolutional Neural Networks were first utilized for skin lesions classification in 2017 by Esteva et al [2], using two classes: benign and malign. Several other subsequent publications dealt with this binary classification using CNNs [1]. However, a binary classification does not fully reflect the reality of skin cancer diagnosis, where multiple types of skin lesions must be considered, therefore, some other recent studies investigated the classification into different types of skin lesions [4] [3] [5] [6] [8].

2 Solution

2.1 General approach

We tried the following different approaches to see which one achieves the best results on skin lesion classification into 7 classes:

1. New ConvNets: two versions of CNN implemented from scratch.
2. Transfer Learning: we decided to proceed with transfer learning because our dataset was considerably small after the deletion of duplicates. Therefore, we realized that a pre-trained network that used millions of images and hundreds of layers can have a huge impact on our accuracy.

- ResNet50: residual network with 50 layers, pre-trained on the ImageNet dataset. Well known as the base model for the transfer learning in image recognition. Trained to identify wide feature representations for wide range of images.
- Mobilenet:
  (a) Lightweight Architecture: depthwise separable convolution where instead of combining all three colour channels and flattening it, it performs one single convolution on each channel and then a 1x1 convolution to combine their outputs, resulting in a total number of computations 8-9 lower than standard convolutions.
  (b) Extensive Dataset: over 1.4 million images and 1000 classes.

2.2 Design

2.2.1 Platform

All the following experiments were performed on Google Collab using Tesla V100-SXM2/Tesla P100-PCIE GPU as well as Tesla T4 for training the models. The following libraries were used in the experiments: PyTorch (for the new convnets models), Keras (for the transfer learning models), Matplotlib, Numpy, Sklearn, Shutil.

2.2.2 Dataset

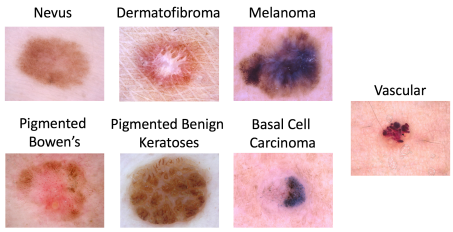

Figure 1: Lesion Categories

We use the MNIST HAM10000 dataset, which consists of 10015 images of skin lesions and a metadata csv file. Each image of lesion was classified into one of the following 7 categories:

- Melanoma: malignant skin cancer.
- Melanocytic nevus: benign type of tumor, however there is a higher risk to contract melanoma.
- Basal cell carcinoma: most common cancerous type of skin lesion.
- Actinic keratosis/Pigmented Bowen's: noncancerous, but when left untreated usually develops into a cancerous skin lesion.

- Benign keratosis: noncancerous and harmless.
- Dermatofibroma: noncancerous and harmless.
- Vascular lesion: may be benign or malignant.

Data Preprocessing: we rearranged the data into subfolders for the classes, this format is necessary for the use of ImageFolder and ImageDataGenerator.

2.2.3 Challenges

Duplicates: we noticed that the dataset contains a large number of instances that were duplicated. We deleted them to avoid the model learning biases towards those duplicated samples. However, that made our data go from 10015 samples to 5514 samples.

Small Sample Size: small training sets complicate having an adequate training. In addition,to avoid making our training set even smaller, for most models we did not use a test set, only a val set. To combat this issue, we used augmentation, with a number of transformations such as RandomAffine, RandomCrop, RandomHorizontalFlip. Additionally, we tested transfer learning.

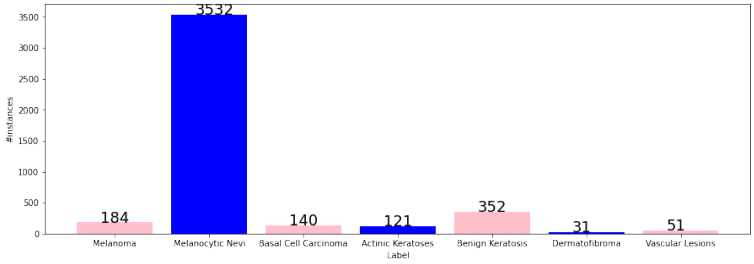

Figure 2: Number of unduplicated instances for each label

Imbalanced Data: as shown above, the dataset is extremely imbalanced. Our biggest complication from this imbalance was the "accuracy paradox" - a situation where the model learns to be "naive" and tends to predict the class with the largest amount of instances in order to achieve a good accuracy. To deal with the imbalance we used weights for transfer learning.

Fluctuating Val Accuracy: since high learning rates and overfitting could be the cause of fluctuating val accuracy, we used kernel_regularizer to deal with both issues.

Overfitting: to solve this issue we used kernel_regularizer and dropout layers. We also tested different amounts of freezing layers for transfer learning to choose the one with minimal signs of overfitting.

2.2.4 ConvNet Version 1

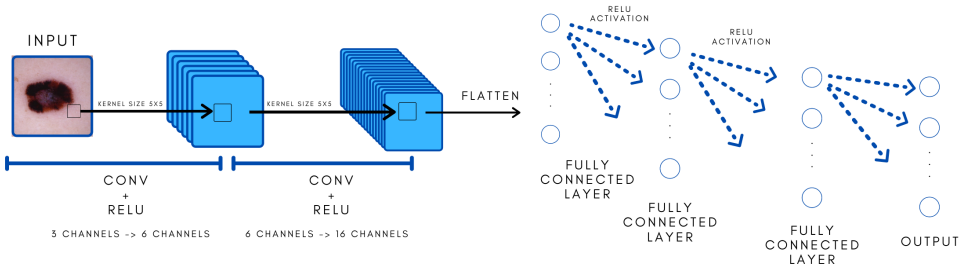

Figure 3: Structure of the ConvNet v1

Average running time after 10 epoch: 650sec

2.2.5 ConvNet Version 2

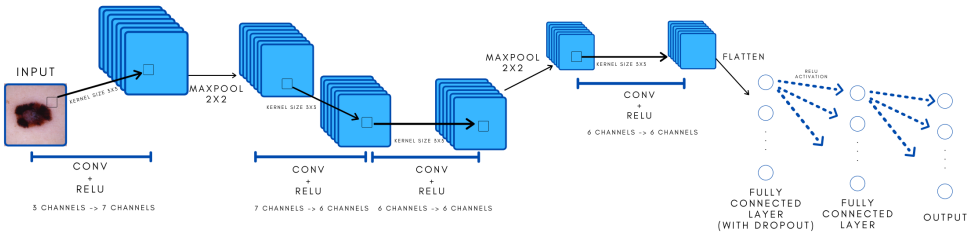

Figure 4: Structure of the ConvNet v2

Average running time after 10 epochs: 670 sec

2.2.6 Transfer Learning: Mobilenet

First we resized all images to be the same as the input layer of MobileNet (224x224). We then removed the last 5 layers and added a dense output layer for 7 classes with a softmax activation function.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenet_1.00_224 (Function (None, 1000)              4253864
_________________________________________________________________
dropout_6 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_6 (Dense)              (None, 7)                 7007
=================================================================
```

Figure 5: Structure of the MobileNet transfer learning model

Average running time with duplicates: 6846sec

Average running time without duplicates: 3410sec

The big difference in average running time, in comparison to the ones of the other models, is mostly due to the use of Tesla T4 for this model, while the rest used Tesla V100-SXM2/Tesla P100-PCIE.

2.2.7 Transfer Learning: ResNet50

The following model uses ResNet50, pretrained on the ImageNet with additional two fully connected layers. The structure of the fully connected layers were added based on the general architecture of the models found online, that used ResNet50 and showed good performance on the dataset we are working with.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 2048)              23587712
_________________________________________________________________
batch_normalization_2 (Batch (None, 2048)              8192
_________________________________________________________________
dropout_2 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               262272
_________________________________________________________________
batch_normalization_3 (Batch (None, 128)               512
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 903
=================================================================
```

Figure 6: Structure of the ResNet50 transfer learning model

Additional batch normalization layers, as well as Dropout layers with rate=0.3 were added before and after the first fully connected layer to prevent model from overfitting. After performing the number of experiments, the number of unfrozen layers in the model was set to last 15 .

Average running time after 10 epochs with LR=0.01: 730 sec

Average running time after 5 epochs with LR=0.001: 1060 sec

Average running time after 5 epochs with LR=0.0001: 1390 sec

3 Experimental results

Splitting of data into the training/validation sets: we chose an 80/20 ratio because our focus is best training results rather than statistical certainty about our results.

Measurement Metrics: categorical accuracy, cross-entropy loss and confusion matrix.

3.0.1 ConvNet Version 1

The model has shown a significant improvement in terms of loss/accuracy in both training and validation in comparison to experiment on the dataset which contains duplicated, which previously showed validation loss of 1.0342 and validation accuracy of 67.0341.

Figure 7: ConvNet v1 training/validation loss/accuracy after epoch 10

3.0.2 ConvNet Version2

The second version of the model has shown performance similar to the ConvNet v1. It is clear that the model is currently underfitting, therefore a more complicated model needs to be implemented.

```
train-loss: 0.6736, train-acc: 80.8660
validation loss: 0.6605, validation acc: 80.0544
```

Figure 8: ConvNet v2 training/validation loss/accuracy after epoch 10

3.0.3 Mobilenet

Our initial tests with transfer learning started with Mobilenet. After dealing with many challenges described in the design section we were excited about achieving good accuracy and loss results. Those were greatly improved mostly with the deletion of the duplicates.

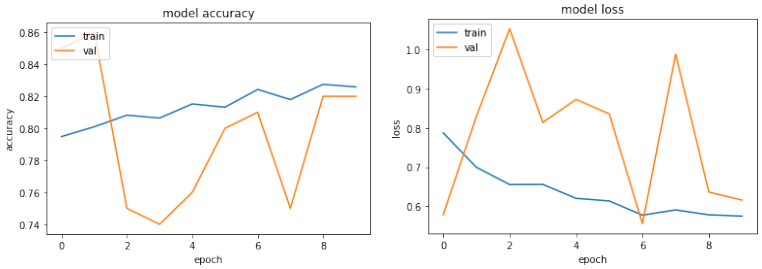

Figure 9: Accuracy and loss for Mobilenet

However, after analysing the predicted classes we realized that the model was naive and only predicting the largest classes in the sample. The problem was alleviated by the use of weights, but it was still not great. After tests we realized that resnet is a better model for transfer learning for our task.

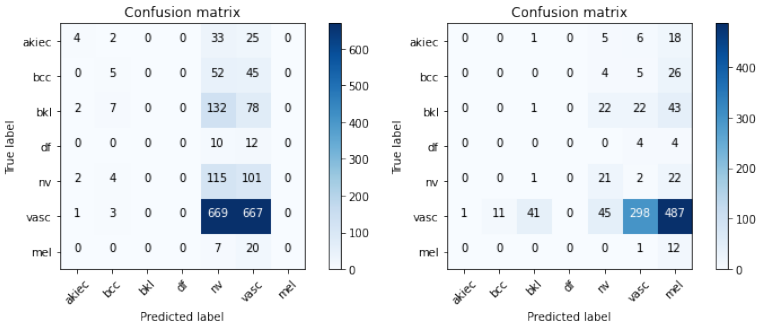

Figure 10: Confusion Matrices Without/With Weights

3.0.4 ResNet50

A number of experiments were performed to tune hyper-parameters such as learning rate and identify how the model is dealing with the data being so imbalanced, to decide if the weights should be assigned to the classes.

First, the model ran with 22 ResNet layers unfrozen, with addition to the fully-connected layers we added. Overall, it was noticed that the model experiences overfitting according to the following results:

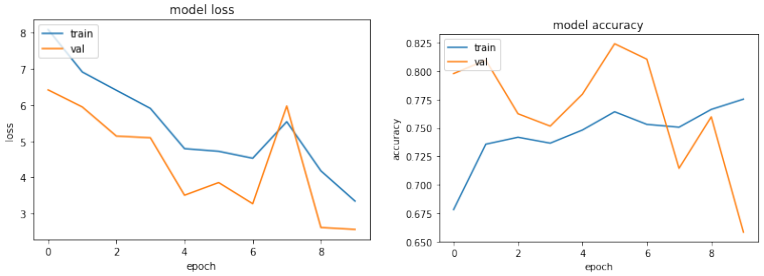

Figure 11: ResNet50 with 22 layers unfrozen training/validation loss/accuracy after 10 epochs

Due to the overfitting, the number of learnable layers was decreased to 15. This version of the model has undergone 4 experiments in which we tested performance of the model with learning rate of: 0.01, 0.001, 0.0001 and with/without using weights of the classes. 0.001 was shown to be an optimal learning rate, by having the best loss/accuracy of 0.68/0.85. Results have shown that models which are not using weights tend to be more "naive" in predicting one class, although they have a good accuracy of 0.855. The confusion matrix for the predictions on the validations set shows that some of the classes are barely predicted, at the same time as class "VASC" is predicted significantly more often, which is the main cause of the high accuracy.

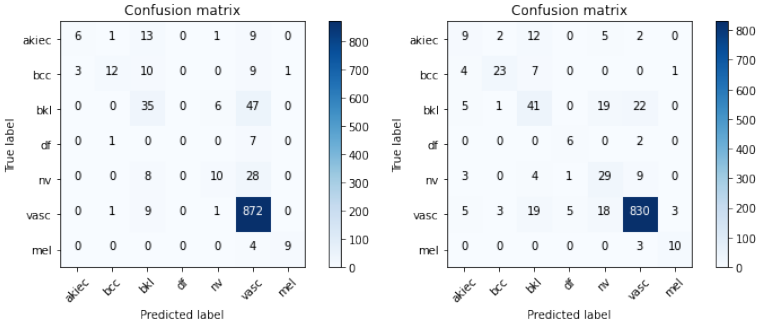

Figure 12: Confusion matrices for ResNet50 without and with weights

After giving each class a weight according to the number of instances in it (with the additional multiplication of the weight of class "VASC" by 5, since it contains the largest amount of instances) the results presented by the confusion matrix changed significantly while the loss/accuracy became 1.34/0.86.

4 Discussion

The performed experiments have highlighted the number of key details regarding the implementation of the model for a dataset. The choice of the architecture of model to perform well on the particular dataset can a main challenge during the construction of the classifier. However, in the case of limited data, even the models that are totally different in their architectures can produce similar results. Data preprocessing has shown to have a significant impact on the performance of the models, especially in situations of limited data. To conclude, we have seen that evaluation of the performance requires the use of different evaluation techniques and metrics.

5 Code

https://colab.research.google.com/drive/1facsG4FiOi9yy2yrCJ6mRg7uykSpgYt1?usp=sharing
http://www.sharelatex.comSomething Linky

References

[1] Roman C.Maron. Systematic outperformance of 112 dermatologists in multiclass skin cancer image classification by convolutional neural networks.

[2] Andre Esteva. Dermatologist-level classification of skin cancer with deep neural networks.

[3] Kiran Pai; Anandi Giridharan. Convolutional neural networks for classifying skin lesions.

[4] Joey Mach. Teaching machines to detect skin cancer.

[5] Aryan Misra. Classifying skin lesions with convolutional neural networks.

[6] Aryan Misra. Classifying skin lesions with convolutional neural networks.

[7] M E Vestergaard. Dermoscopy compared with naked eye examination for the diagnosis of primary melanoma: a meta-analysis of studies performed in a clinical setting.

[8] Xinrui Zhuang. Skin lesion classification.