**Intracranial Hemorrhage Detection**

**Using Neural Networks**

***Team:*** *Ghiga Claudiu-Alexandru, Lionte Bogdan,*

*Nistor Șerban, Tatu Georgian-Adrian*

1. **Problem description**

Intracranial hemorrhage, bleeding that occurs inside the cranium, is a serious health problem requiring rapid and often intensive medical treatment. The process of detecting the hemorrhage presence, including the detection of its subtype it is a challenging process and often time consuming.

Therefore, the challenge imposed by this task is to build an algorithm that can detect acute intracranial hemorrhage and its subtypes: intraparenchymal, intraventricular, subarachnoid, subdural and epidural. (Figure 1)

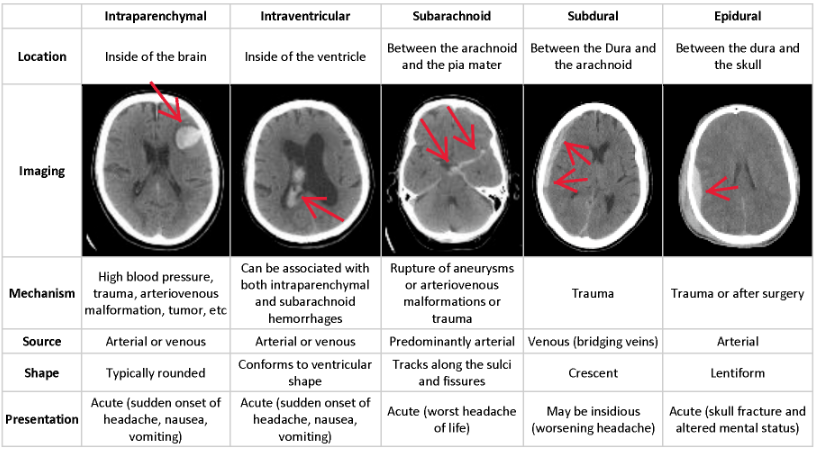
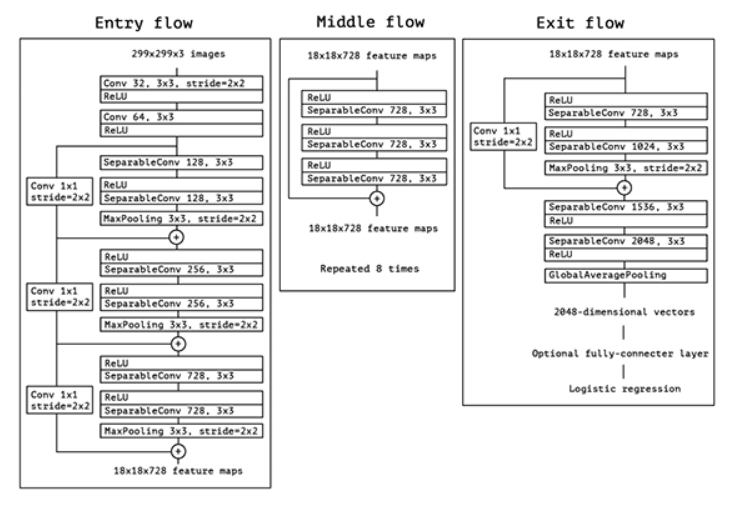


Figure 1

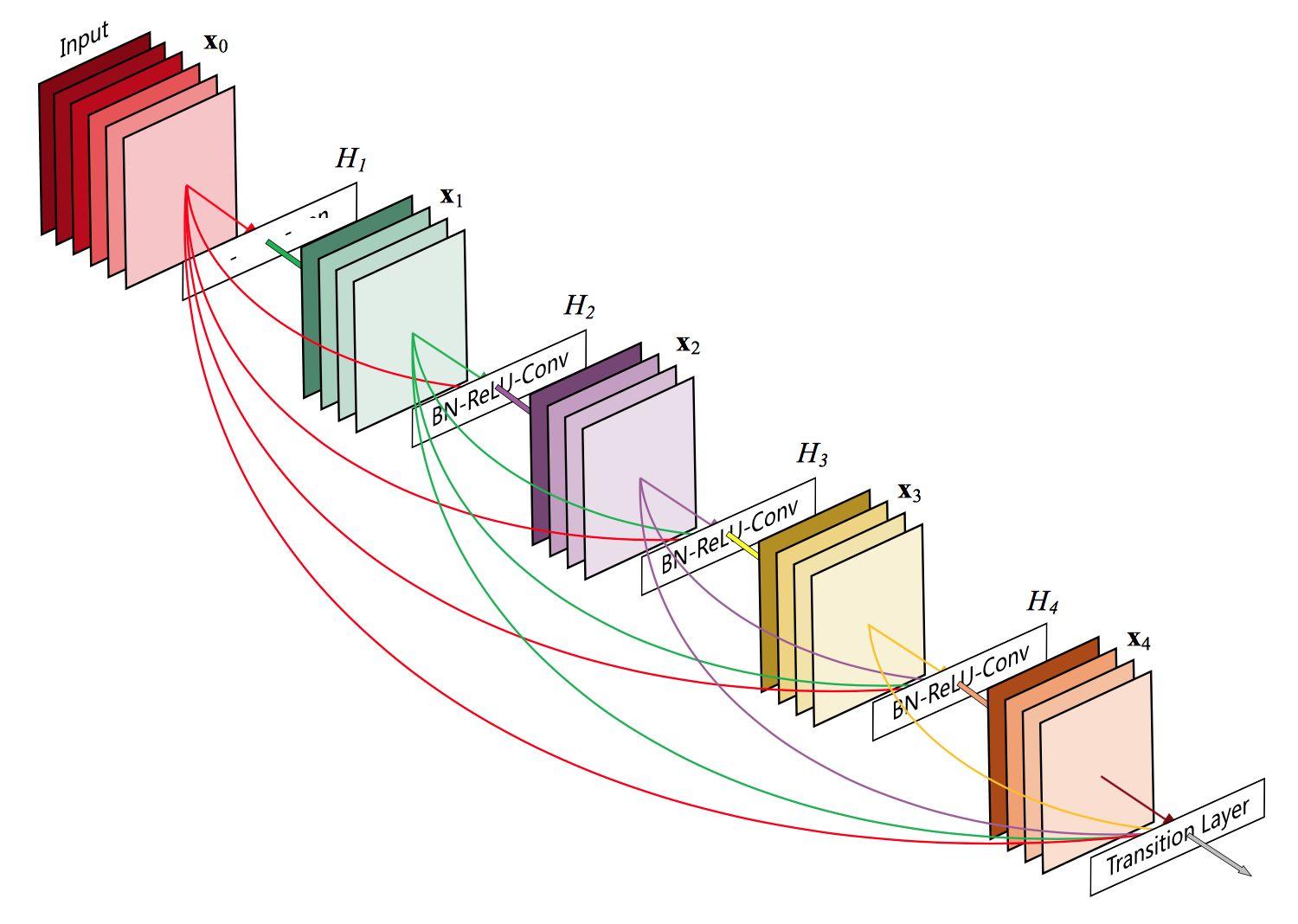
The dataset for this problem is provided by the Radiological Society of North America (RSNA®) in collaboration with members of the American Society of Neuroradiology and MD.ai. and consists of 752.803 dicom files available for training and another 121.232 dicom images usable for the testing of the proposed algorithm. Dicom images are used in modern radiological imaging and they provide besides the pixel information additional details for pacient position, pacient uid, pacient orientation, rescale intercept, rescale slope, etc.

The performance of the proposed mdel is evaluated using a weighted multi-label logarithmic loss. Each hemorrhage sub-type is its own row for every image, and you are expected to predict a probability for that sub-type of hemorrhage. There is also an any label, which indicates that a hemorrhage of ANY kind exists in the image. The any label is weighted more highly than specific hemorrhage sub-types.

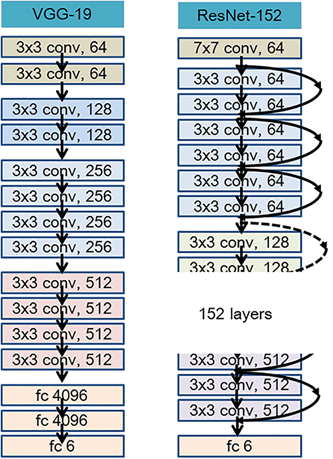
1. **State of the art**
   1. *Xception***:** [**https://arxiv.org/abs/1610.02357**](https://arxiv.org/abs/1610.02357)

****

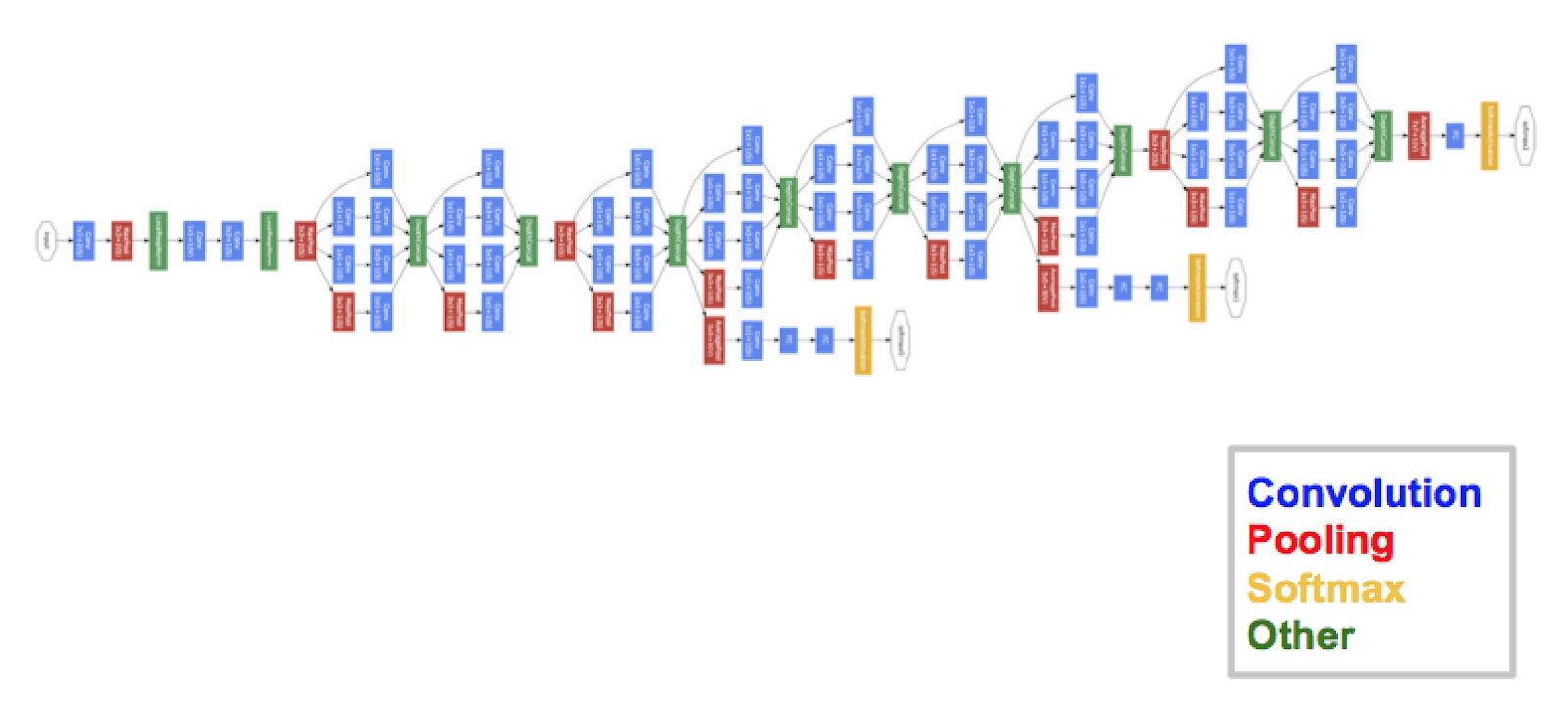
* 0.790 accuracy on the ImageNet data set
* The model features 126 layers with 22 million weights
* It accepts default input size of 299x299, but not less than 71x71
  1. *DenseNet***:** [**https://arxiv.org/abs/1608.06993**](https://arxiv.org/abs/1608.06993)

****

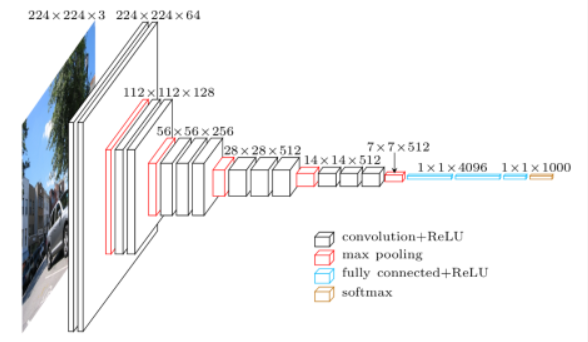
* There are 4 architectures available, the best accuracy being obtained by DenseNet201
* The highest accuracy is 0.773
* The model features 201 layers with 20 million weights
  1. *ResNet***:** [**https://arxiv.org/abs/1512.03385**](https://arxiv.org/abs/1512.03385)

****

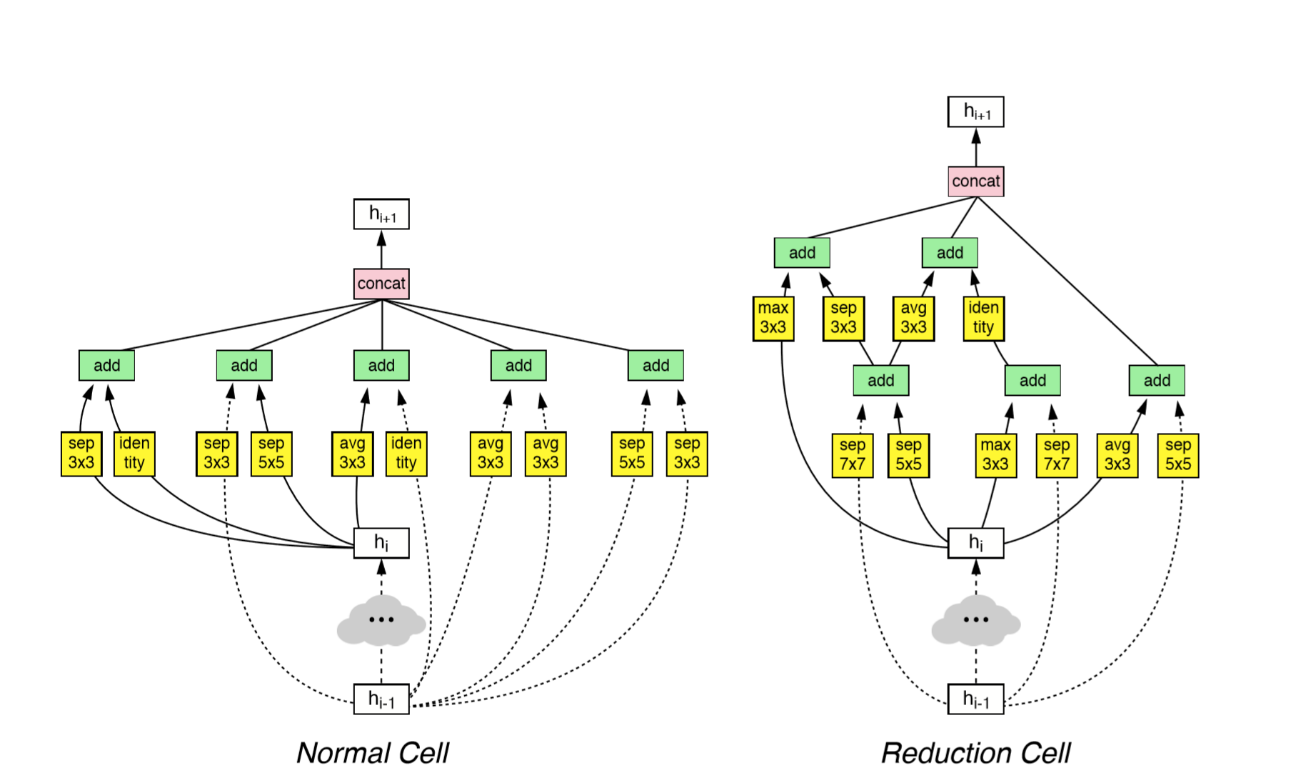
* There are 5 architectures available, the best accuracy being obtained by ResNet152V2
* The highest accuracy is 0.780
* The model has 60 million weights
  1. *Inception***:** [**https://arxiv.org/abs/1512.00567**](https://arxiv.org/abs/1512.00567)

****

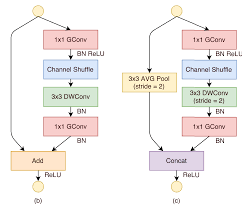
* Two architectures are available: InceptionV3 and InceptionResNetV2
* InceptionV3 presents 159 layers with 22 million weights
* InceptionV3 achieves an accuracy of 0.779
* InceptionResNetV2 presents 572 layers with 55 million weights
* InceptionResNetV2 achieves an accuracy of 0.803
  1. *VGG*: <https://arxiv.org/abs/1409.1556>



* 2 architectures are available: VGG16 and VGG19
* Both architectures support 224x224 default input
* VGG16 has an accuracy of 0.715, and VGG19 has an accuracy of 0.727
* VGG16 has 23 layers with 138 million weights
* VGG19 has 26 layers with 143 million weights
* Very hard to train
* Weights are high in storage (500mb +)
  1. *NASNet*: <https://arxiv.org/abs/1707.07012>

****

* There are two architectures, Mobile and Large
* Mobile has 5 million weights, and Large has 88 million weights
* Mobile reaches an accuracy of 0.744, and Large reaches an accuracy of 0.825
  1. *MobileNetV2*: <https://arxiv.org/abs/1801.04381>

****

* The model has 88 layers with 3.5 million weights
* It reaches an accuracy of 0.713

1. **Our Solution**

Our proposed system for solving this problem can be divided in three core parts. A preprocessing system, that applies a series of transformations to the initial images, an augmentation step that is responsible with extending the current dataset by using certain imaging techinques.

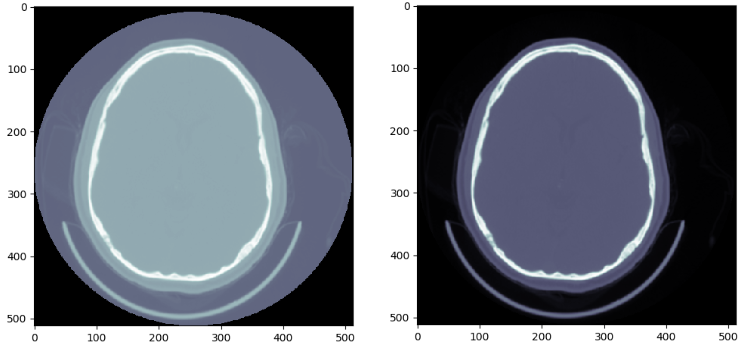
Last but not least, we designed a neural system that, with the help of the previous two systems aims to predict if a pacient has hemorrhage and of course the subtype of hemorrhage if that’s the case.

**I. Preprocessing chain**

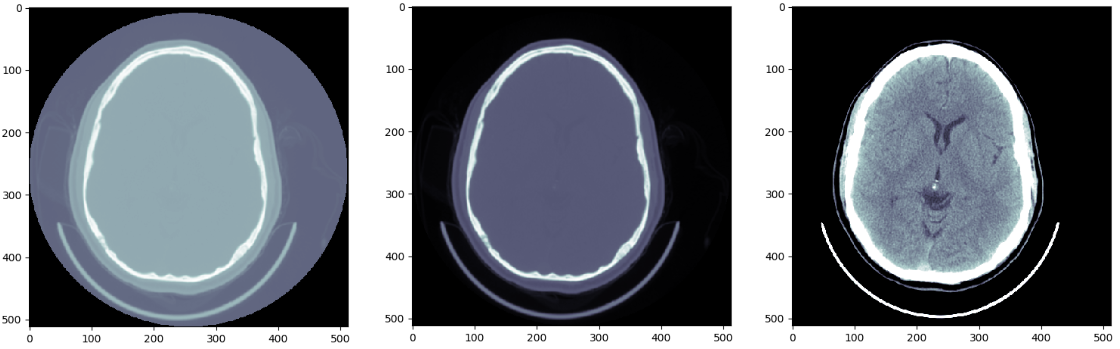
The dataset contains almost only 512x512 images with a few exceptions. Because these exceptions exist, the first step in our preprocessing chain is to make sure that all images are the same size and we do that by resizing the images that not correspond to these dimensions.

Secondly, the most important step from this preprocessing chain, consists in applying a hounsfield transformation to the images. The unit of measurement in CT scans is the Hounsfield Unit (HU), which is a measure of radiodensity. CT scanners are carefully calibrated to accurately measure this. By default however, the values from the pixel arrays are not in this unit.

The hounsfield transformation that we apply does just that, by multiplying with the rescale slope and adding the intercept, which we can get from the metadata of the files, we shift the values domain to HU domain. Below is an example of an image before applying this type of transformation, and after:



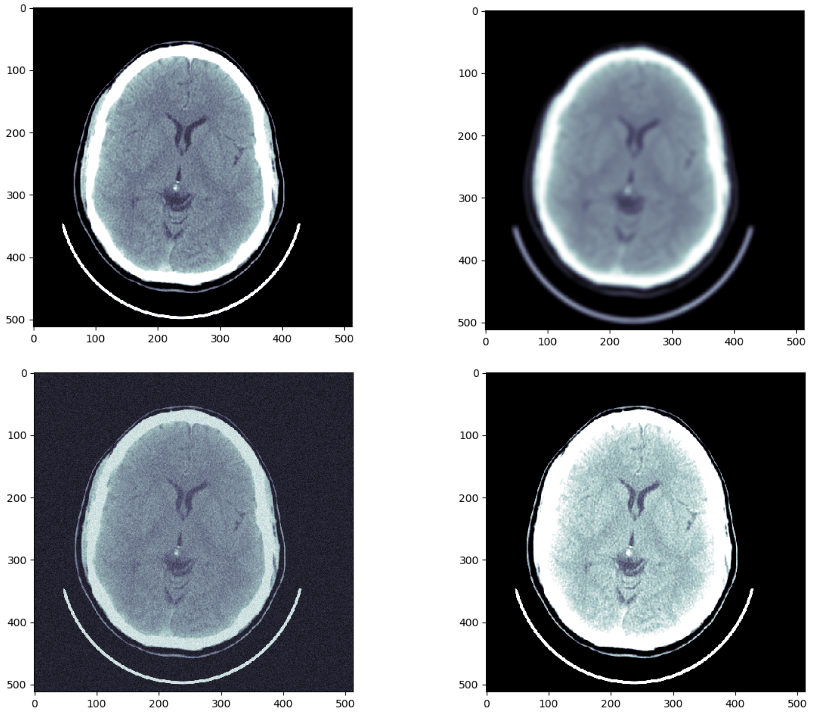
And last but not least, we use a windowing procedure on the resulted image from the previous step. There are multiple windows from which we could choose, but we chose to use a soft tissue window to evidentiate the brain tissue more clearly and so to provide a higher chance to our system, of predicting subdural hemorrhage, which is located on the margins of the brain and it’s hardest one to notice. Below is presented an image that went through all the mentioned preprocessing steps:



**II. Data augmentation**

If we take a look at the distribution of the images, regarding the classes and subclasses, we observe something interesting. The fact that the images that present hemorrhage are only 5% from the total number of the images from the training set and we have to deal with an unbalanced dataset. With this in mind, we apply a probability weighted augmentation to the undersampled class.

We applied three different augmentation procedures: blur, gaussian noise and an increase in brightness. The result of these procedures is presented below:



In the top left corner is the original image, in the top right is the blurred version of the same image, in the bottom right is the brightened one and in the bottom left it’s the image with gaussian noise.

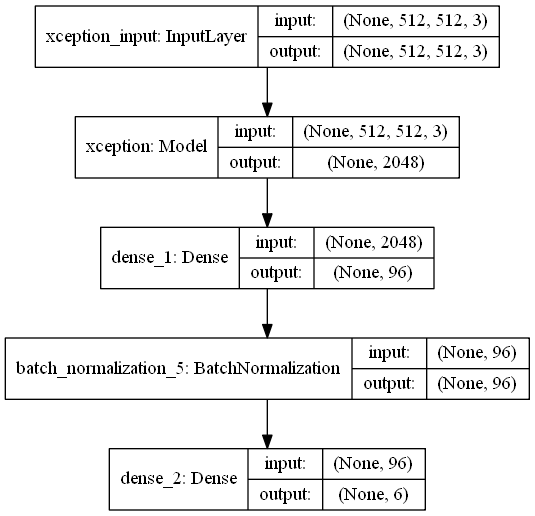
**III. Neural system architecture**

Our neural network approach is somewhat complex, given the difficulty of the problem we had to solve. We use not one, but two different neural networks that aid us in predicting whether a pacient has hemorrhage or not and also which subtype of hemorrhage does it have if that’s the case.

Before talking about the neural networks that we used, an important step without whom, the second neural network would have been useless it’s this: we observed that in this dataset, groups of images belong to the same pacient.

We also noticed that the size of these groups of images vary from 20 to about 100 and they can be arranged in a specific order, by using the image positioning from the metadata. By doing this, we have obtained scanning sequences of all the pacients that are present in the dataset.

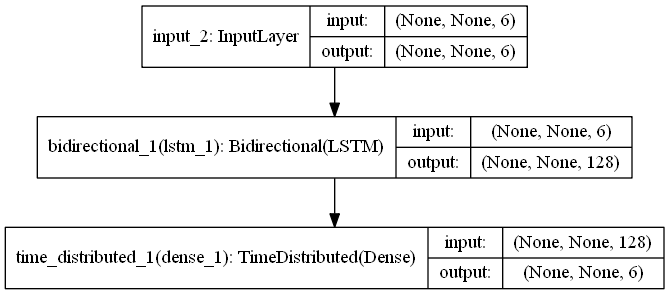
Now, for the first neural network we use a pretrained convolutional neural network, called Xception and we fine tune it by adding a leaky relu and a sigmoid layer on top of it. We choose to use leaky relu and not ordinary relu activation to avoid the dying relu problem, that can arise in sparse neural network architectures.



The sigmoid layer will be responsible with classifying the input in 6 classes which aren’t mutual exclusive: any, epidural, intraparenchymal, intraventricular, subarachnoid and subdural.

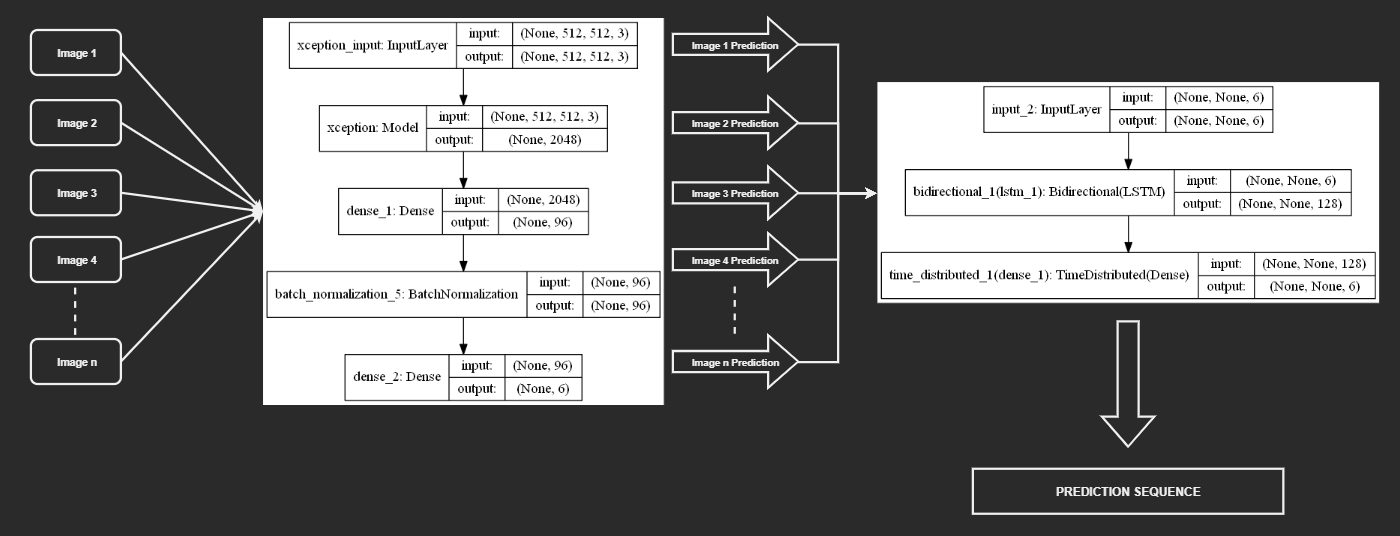
The training phase of this neural network is done on every image, with no grouping by pacient applied. The interesting part comes when predicting the output: the images are grouped by pacient UID in sequences. So, the predictions given by this first neural network will be sequences of non-mutual exclusive probabilities.

These probabilities are fed to the second neural network.



An important attribute of this model is the fact that it accepts sequences of variable length. We use a bidirectional LSTM layer that can look at the probabilities sequences from both sides. As the final layer for this network we use the same type of layer as the one in the precedent architecture, only that this time, the layer is wrapped in a TimeDistributed type of layer, so it can output multiple sequences of probabilities.

The idea behind this architecture is to use the second neural network to fine-tune the probabilities outputed by the first neural network. The architecture for the whole neural system can be viewed below:



1. **Results obtained**

The first try at solving this problem was done using an architecture that consists of two models. One model was responsible with predicting whether an image presents hemorrhage or not (binary classification), and if the response was positive, the second model’s task was to predict the subtype of hemorrhage present in that image (categorical-classification).

We also chose a base model, which was already trained on the ImageNet dataset, for both the binary model and the categorical one. We made the choice by taking into account both the accuracy that it can provide on our problem but also the loss that it obtained (Figure 2).

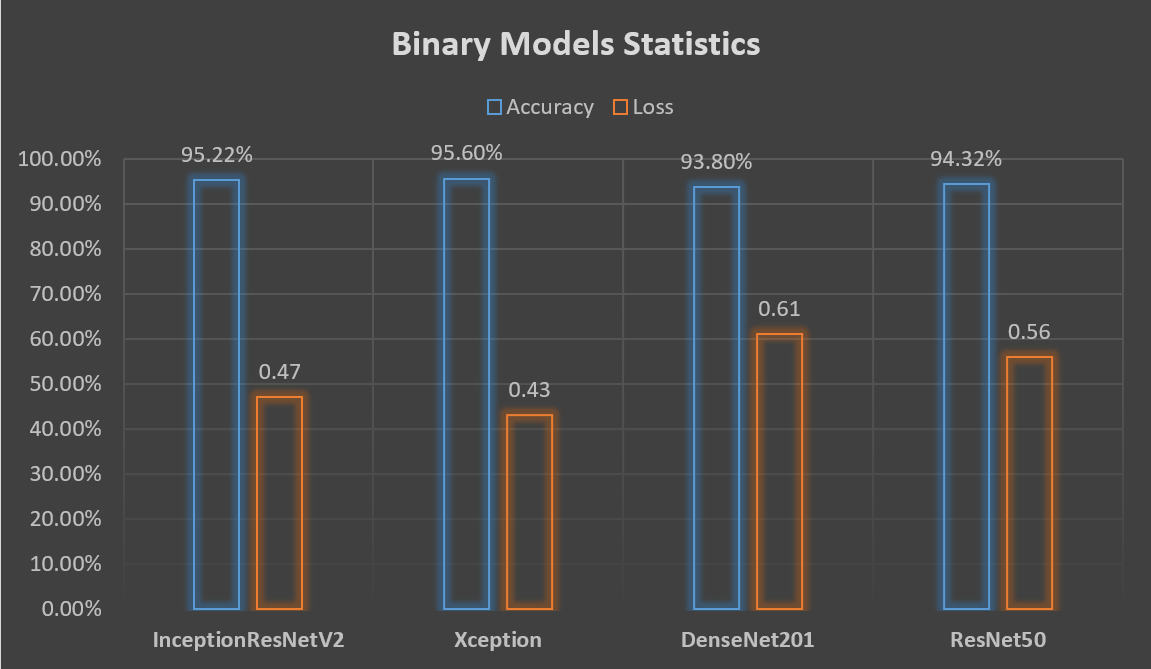
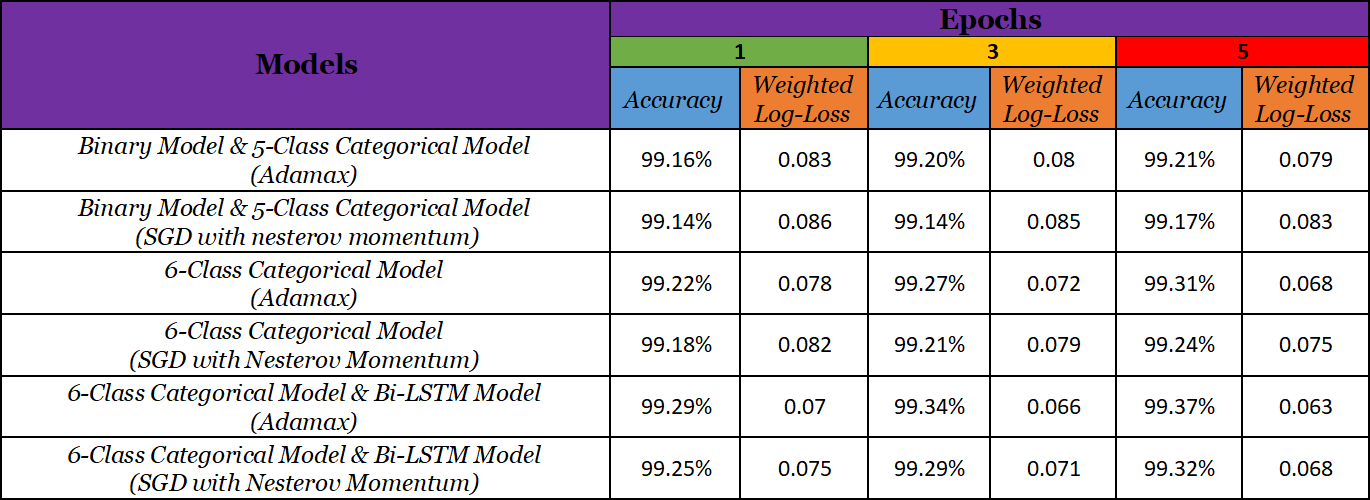


Figure 2

The results obtained by using this architecture were good, but not extraordinary, the weighted logaritmic loss was only at 0.083. After this result, we tried to see if we can obtain an improvement by training a model on all six classes all at once. The results in this case were astonishing, the model obtaining a loss of 0.078 which is not a big improvement over the previous result, but it was still an improvement.

We began from there and we stacked on top of that model a Bidirectional LSTM network which accepted as the input, the predictions from the previous model, and it’s purpose was to fine-tune the probabilities, having the feature of sequences.

The things went well with this type of architecture, and it seemed that our intuition was right regarding the power that a recurrent network can give to our initial model in this context. After training this type of architecture and after small fine tuning of the hyper parameters we achieved a loss of 0.063. A table with all of these results is available below (Figure 3).

****

1. **Comparison with other methods**

In this section of the paper, we will compare our solving method with the top three methods from the IHD Kaggle competition. We will talk about three aspects: preprocessing /augmentation procedures, model complexity and last but not least we will compare the final results obtained.

1. **Conclusion and future work**

There are two major conclusions to this paper. The first one is that as we can see from the above graphs, a model trained on all six classes performs better that two sepparate models, one that is trained to predict if a hemorrhage is present and another that predicts the subtype if it’s the case.

About the second conclusion, we can clearly see that if we integrate in the whole system a recurrent model and if we can construct from the provided data, meaningful sequences, the prediction power of the whole architecture can and should increase considerably. As we have seen, both the accuracy and the weighted log loss got a noticeable improvement.

As for the future work part, improvements can be made on the preprocessing part of the system, where smarter processing techniques can be applied and also on the augmentation part, where if we apply more complex augmentation procedures we can achieve better results.