

behold.ai – Coding Challenge



Author: Drew Afromsky
Email: daa2162@columbia.edu
Date: 02/05/2020

Abstract.

The challenge presented in this coding assessment is a multi-label image classification problem. This is a classification task where each input sample (i.e. brain CT images) can be assigned multiple labels. For instance, a given image may contain both an epidural and intraparenchymal hemorrhage and should be annotated with 2 labels corresponding to each ICH type. In this challenge, there are 3 possible labels (intraparenchymal, epidural, subarachnoid) that can be assigned to an image (CT image).

Two different network architectures were introduced in this challenge. Specifically, to address the task of this challenge transfer learning using a pre-trained convolutional neural network (CNN) was implemented. The feature extractor/backbone network architectures explored include Xception, InceptionV3, ResNet50, VGG16, and VGG19, as these are the image-classification models pre-trained on ImageNet that are available as part of Keras. The top layers were trained as classifiers and used to make predictions on the test images. The predictions were recorded in two csv files, one providing binary predictions for each class label and one with class-likelihood probabilities.

Python 3 and Tensorflow 2.0 along with machine learning packages and libraries (numpy, scipy, pandas, opencv, etc.) were used for acquiring and structuring the data, selecting deep learning models, and evaluating these models. For ease of use by the team at Behold.ai, the notebooks for training and model evaluation can be ran in Google Colab and Kaggle notebooks/kernels once the train/test images have been uploaded and/or unzipped into their corresponding folders. NVIDIA K80 GPU was used in Google Colab and Kaggle kernels to train the neural networks. To train multiple networks more efficiently, multiple models were implemented in parallel in each of these online notebook environments (i.e. one network in Google Colab and one in Kaggle kernel with slightly different hyperparameters).

Introduction.

Hemorrhage in the head (intracranial hemorrhage) is a relatively common condition that has many causes ranging from trauma, stroke, aneurysm, vascular malformations, high blood pressure, illicit drugs and blood clotting disorders. The neurologic consequences also vary extensively depending upon the size, type of hemorrhage and location ranging from headache to death.

Intracranial hemorrhage, bleeding that occurs inside the cranium, is a serious health problem requiring rapid and often intensive medical treatment. For example, intracranial hemorrhages account for approximately 10% of strokes in the U.S., where stroke is the fifth-leading cause of death. Identifying the location and type of any hemorrhage present is a critical step in treating the patient. Diagnosis requires an urgent procedure. When a patient shows acute neurological symptoms such as severe headache or loss of consciousness, highly trained specialists review medical images of the patient's cranium to look for the presence, location and type of hemorrhage. The role of the Radiologist is to detect the hemorrhage, characterize the hemorrhage subtype, its size and to determine if the hemorrhage might be jeopardizing critical areas of the brain that might require immediate surgery. The process is complicated and often time consuming. Deep learning has the potential to help the medical community identify the presence, location and type of hemorrhage in order to quickly and effectively treat affected patients.

While all acute (i.e. new) hemorrhages appear dense (i.e. white) on computed tomography (CT), the primary imaging features that help Radiologists determine the subtype of hemorrhage are the location, shape and proximity to other structures.

Intraparenchymal hemorrhage is blood that is located completely within the brain itself; intraventricular or subarachnoid hemorrhage is blood that has leaked into the spaces of the brain that normally contain cerebrospinal fluid (the ventricles or subarachnoid cisterns). Extra-axial hemorrhages are blood that collects in the tissue coverings that surround the brain (e.g. subdural or epidural subtypes). Patients may exhibit more than one type of cerebral hemorrhage, which may appear on the same image. While small hemorrhages are less morbid than large hemorrhages typically, even a small hemorrhage can lead to death because it is an indicator of another type of serious abnormality (e.g. cerebral aneurysm).


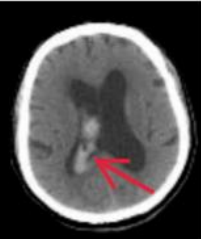

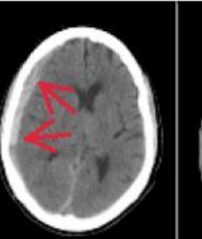

	Intraparenchymal	Intraventricular	Subarachnoid	Subdural	Epidural
Location	Inside of the brain	Inside of the ventricle	Between the arachnoid and the pia mater	Between the Dura and the arachnoid	Between the dura and the skull
Imaging					
Mechanism	High blood pressure, trauma, arteriovenous malformation, tumor, etc	Can be associated with both intraparenchymal and subarachnoid hemorrhages	Rupture of aneurysms or arteriovenous malformations or trauma	Trauma	Trauma or after surgery
Source	Arterial or venous	Arterial or venous	Predominantly arterial	Venous (bridging veins)	Arterial
Shape	Typically rounded	Conforms to ventricular shape	Tracks along the sulci and fissures	Crescent	Lentiform
Presentation	Acute (sudden onset of headache, nausea, vomiting)	Acute (sudden onset of headache, nausea, vomiting)	Acute (worst headache of life)	May be insidious (worsening headache)	Acute (skull fracture and altered mental status)

Figure 1. Examples of the different sub-types of intracranial hemorrhages. Image taken from [Kaggle](#)

Methods.

Datasets

All data used in this challenge was provided by Behold.ai. The data has been extracted from a Kaggle competition and includes 8,542 training images and their corresponding labels, along with 4,019 testing images. Each image is either healthy (no ICH), or contains at least one ICH sub-type.

Data Analysis

In order to get a better idea of the dataset, data analysis was performed prior to any actual machine learning. The goal of data analysis is to better understand the datasets and discover trends or distributions in the data. For example, data analysis tools can reveal class imbalance in the dataset, which is important to take into consideration when evaluating a model and deciding upon the appropriate performance metrics.

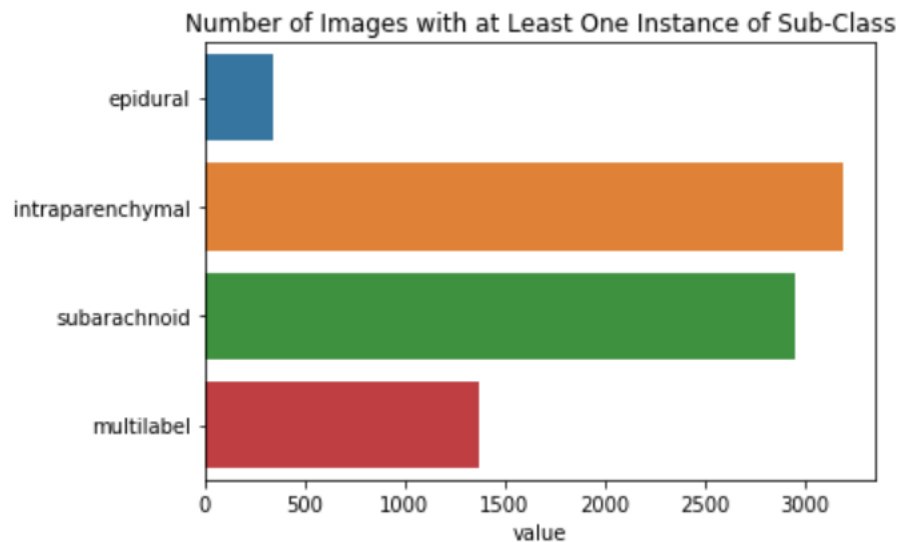


Figure 2. Distribution of classes among the training images, where a single image may have more than one class label

Feature Extraction and Transfer Learning

Due to the time constraint for submission, along with the scarcity of data, transfer learning using a pre-trained convolutional neural network was implemented to perform multi-label image classification. A pre-trained network is a saved network previously trained on a large dataset, typically on a large-scale image-classification task (Imagenet in our case). If the original dataset is large enough and general enough, then the spatial hierarchy of features learned by the pretrained network can act as a general model of the visual world, and its features can be useful for other computer vision tasks, despite involving different images and classes. Portability of learned features across different problems is a key advantage of deep learning.

Feature extraction using the aforementioned backbone architectures consists of using the representations learned by the previous networks to extract interesting and hopefully relevant features from new data samples. These features run through a new classifier trained from scratch. In Keras, the top layers correspond to the final consecutive dense layers. If `include_top = 'False'`, then we exclude these dense layers and the network's outputs will be the high-level feature maps/bottleneck features of the last convolutional/max-pooling block. For the specific problem at hand, the last layer activation function is a sigmoid and the corresponding loss function is binary cross entropy.

During transfer learning, we add a densely connected classifier on top of the convolutional base. Before compiling and training the model we must freeze the convolutional

base, which means preventing weights of a layer or set of layers from being updated during training.

Data Augmentation

Image data augmentation is a popular technique to represent the same image in a different form. For example, a common image augmentation technique is flipping vertically or horizontally, where the image appears to be "mirrored" or "flipped" along an axis after this augmentation is completed. More specifically, image augmentation is regularly used to increase the size of a dataset that is later used for training algorithms utilizing deep learning. In computer vision applications using deep learning, these algorithms tend to be data hungry, requiring many example images to learn from. Data or image augmentation is an effective way to increase the size of a dataset while maintaining the same kind or type of image examples the algorithm learns from. Data augmentation was applied to combat overfitting. This approach generates more training data from the existing samples via random transformations.

Data augmentation was done at training time using data image generator in Keras, due to the storage space limitations for augmenting thousands of images that would result in thousands of more images. Test time data augmentation was not applied. Augmentation techniques including horizontal and vertical flipping, rescaling, and rotation were used.

Model Selection

In terms of model selection, two different backbone architectures were explored – ResNet50 and InceptionV3. Network backbone architecture that was most successful was InceptionV3. The binary cross entropy (BCE) loss was chosen as the loss function and sigmoid was used as the appropriate activation function in the last layer. The BCE loss was optimized using stochastic gradient descent (SGD) with a Nesterov momentum of 0.9 for the ResNet50 model and two different optimizers were used for the InceptionV3 module – ADAM and SGD. Since the model has already been through a first training phase on a similar dataset and is probably close to convergence already, a smaller learning rate was used for the fine-tuning phase. The learning rate for the best Inception network was set to 1E-5 and the ADAM optimizer was used as the optimization function. The networks were set to be trained for 100 epochs assuming no early stopping had halted training at an earlier epoch.

```
Model: "sequential"
-----
Layer (type)                Output Shape              Param #
-----
keras_layer (KerasLayer)    multiple                  21802784
-----
batch_normalization (Batch Normalization) multiple                  8192
-----
dropout (Dropout)           multiple                  0
-----
dense (Dense)                multiple                  6147
-----
Total params: 21,817,123
Trainable params: 10,243
Non-trainable params: 21,806,880
-----
```

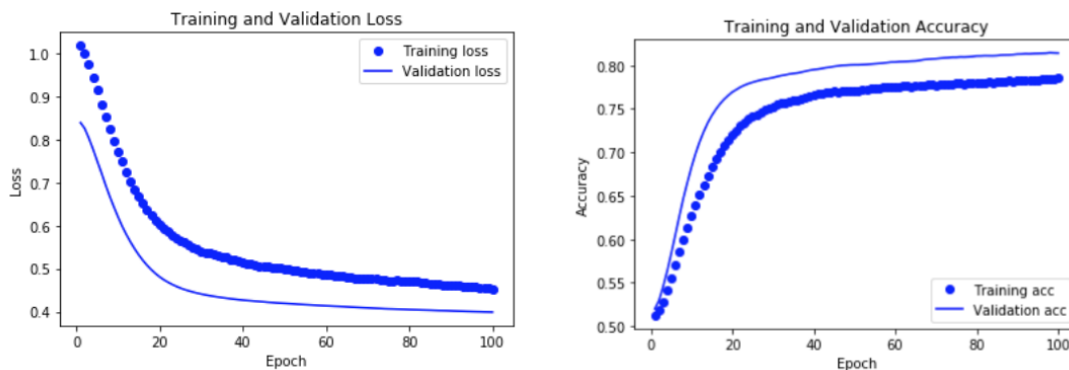
Figure 3. Model Summary – Using Inception V3 as backbone/feature extractor with batch normalization, dropout, and a densely connected layer (classifier) on top.

Regularization

Regularization methods are commonly deployed in machine learning models to avoid overfitting. Popular regularization techniques include L1/L2, early stopping, batch normalization, and dropout. In this challenge early stopping, batch normalization, dropout, and L2 regularization were implemented to combat overfitting. The patience for early stopping was set to 10% of the number of epochs, therefore the patience = 10 during training. L2 regularization factor was set to 0.0001.

Results.

The metrics used to assess the performance of the models include training and validation loss. Based on the training-validation accuracy curves, it looks like the model could have been trained for longer (more epochs) as the validation accuracy seemed to be continuing to increase while the validation loss also showed a downwards trend. The plots below suggest the model has underfit the data and should be trained for longer.



Best Training Accuracy: 78.72%
Best Validation Accuracy: 81.70%

Discussion.

Future areas for exploration include automating the hyperparameter optimization, looking into modifying the network architecture by adding more layers or making the layers wider (more units per layer), accumulating more data or generating new data generative adversarial networks – GANs (as neural networks tend to perform better with more data), class-activation mapping (CAM) for model interpretation, image segmentation for identification/localization of the hemorrhages in the images, trying out different feature extractors such as EfficientNetB2, ResNext, VGG-16/19, Inception ResNet V2, model ensembling – pooling together the predictions of a set of different models to produce better predictions. Ensembling relies on different good models trained independently are likely good for different reasons and each model looks at slightly different aspects of the data to make its predictions.