# ED6001: Medical Image Analysis — Project Report

# Intracranial Hemorrhage Detection

Anoubhav Agarwaal (BE16B002) Rudram (ED16B024)

## 1 Objective

The objective of this project is to perform **multi-label image classification** on a medical image dataset using popular deep learning architectures. We detect **acute intracranial hemorrhage** and its subtypes. The dataset is provided by the Radiological Society of North America(RSNA).

## 2 Introduction

Intracranial hemorrhage, bleeding that occurs inside the cranium, is a severe 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. 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.[1]

There are **five hemorrhage subtypes**: Intraparenchymal, Intraventricular, Subarachnoid, Subdural, and Epidural (refer fig.1). **Patients may exhibit more than one type of cerebral hemorrhage**, which may appear in the same image.

| | 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 |

Figure 1: Intracranial hemorrhage subtypes. [2]

While all acute (or new) hemorrhages appear dense (or 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**.[2]

We build **deep learning algorithms** to detect acute intracranial hemorrhage and its five subtypes. The following section covers the methods applied in developing the deep learning solution pipeline. This includes the steps involved in data cleaning and pre-processing, data augmentation, model training, and model evaluation. For each image ID, only one image is present. Thus, the dataset only contains a **2D slice per patient** and no 3D volume data.

# 3 Methodology

## 3.1 Data description

The dataset is hosted on Kaggle. The head CT scans are provided in **DICOM format**. The DICOM images contain the associated metadata.

The training data is provided as a set of *image ID*'s and multiple labels, one for each of five sub-types of hemorrhage, plus an additional label for *any*, which should always be true if any of the sub-type labels are true. There is also a target column, *Label*, indicating the **probability of whether that type of hemorrhage exists** in the indicated image. The size of the dataset is $\approx$ **180GB**.

## 3.2 Data cleaning and pre-processing

The following steps were performed in chronological order:

1. **Removal of duplicates** (for an image ID) in the train set.
2. **Upsampling of Epidural subtype** by 8 times (by repetitive concatenation of Epidural-positive samples to the dataset). It was observed that the training data consisted of substantial **class imbalance**. The percentage of positive samples in the dataset:

   - any: 14.34%
   - Epidural: **0.42%**
   - Intraparenchymal: 4.8%
   - Subdural: 6.27%
   - Subarachnoid: 4.74%
   - Intraventricular: 3.48%

   After upsampling the Epidural class (by 8 times),

   - any: 17.26%
   - Epidural: **3.34%**
   - Intraparenchymal: 5.39%
   - Subdural: 6.94%
   - Subarachnoid: 5.27%
   - Intraventricular: 3.69%

   **Note:**

   (A) The percentage of positive samples for other subtypes also went up as the images can have more than one cerebral hemorrhage. Thus, some upsampling was observed in other subtypes, as well.

   (B) The **% of negative samples**, i.e., those who don't contain *any* hemorrhage subtypes, is **82.74%**. In hindsight, the number of negative DICOM images could have been undersampled to facilitate learning in our models. In the experiments below, we have a **1:4.8 ratio of positive samples to negative samples**. This should have been reduced to 1:2.

3. **The total size of the DICOM image dataset is ≈ 180 GB**. It was not feasible to work with such large amounts of data for the project due to limitations in memory capacity and RAM of the machine in use. Thus, using the *MultilabelStratifiedShuffleSplit* function provided by the **iterative-stratification package**, we obtained **10% of the original data, stratified on the multi-labels**. This ensured that the distribution of the labels in the new dataset remained the same as the original. The distribution (i.e., the percentage of positive samples for each hemorrhage subtype) of the new dataset is the same as above.
4. This new dataset was further split into a **training and validation set** using the *MultilabelStratifiedShuffleSplit* function. A **90-10 split** was performed yielding, **65859 images in training**, and **11623 images in the validation set**.

## 3.3 Windowing operation

**Windowing**, also known as **grey-level mapping**, contrast stretching, histogram modification or **contrast enhancement** is the process in which the CT image greyscale component of an image is manipulated via the CT numbers; doing this changes the appearance of the picture to **highlight particular structures**(such as the brain or soft tissue).

The **brightness** of the image is, adjusted via the **window level** ($WL$). The **contrast** is adjusted via the **window width** ($WW$). The $WL$ is the midpoint of the range of the CT numbers displayed. The $WW$ is the measure of the range of CT numbers that an image contains. When presented with a $WW$ and $WL$ one can calculate the **upper and lower grey levels**,i.e., values over $x$ will be white, and values below $y$ will be black. [3]

- the upper grey level ($x$) is calculated via $WL + (WW \div 2)$
- the lower grey level ($y$) is calculated via $WL - (WW \div 2)$

Using windows, we can highlight and emphasize specific voxels. There are at least **5 windows** that a radiologist goes through for each scan! These are:

1. **Brain Matter:** WW:80 ; WL:40
2. **Blood/subdural:** WW:130-300 ; WL:50-100
3. **Soft tissue:** WW:350–400 ; WL:20–60
4. Bone: WW:2800 ; WL:600
5. Grey-white differentiation: WW:8 ; WL:32. [4]

Three types of windows were used to focus, and each of them was assigned to a channel. Thus, while training, we construct images on the fly (with windows as channels) using a data generator.

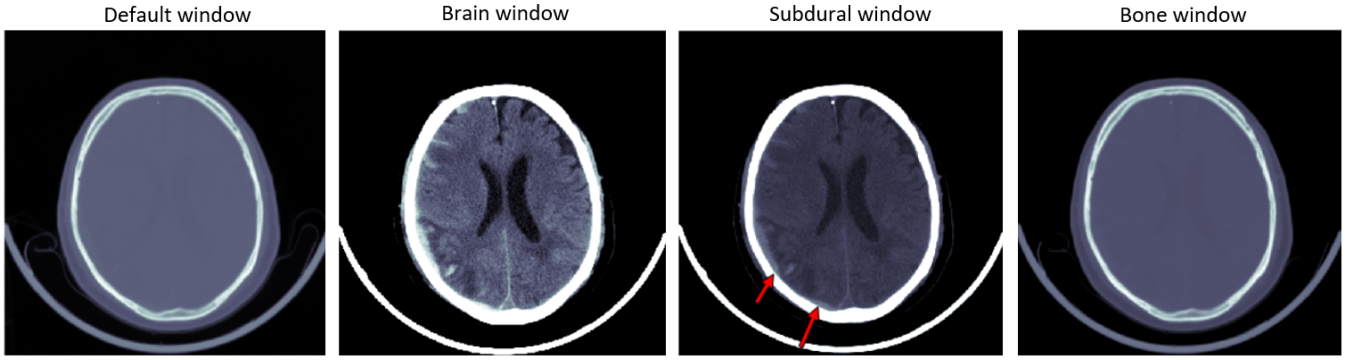



Figure 2: Different windows assigned as channels of the image capture distinct information.

## 3.4 Network Architectures

Table 1: Performance results on ImageNet for models tested.[5, 6]

| Model | Top-1 Acc. | Top-5 Acc. | #Params |
|---|---|---|---|
| VGG16 | 0.713 | 0.901 | 138 M |
| ResNet50 | 0.749 | 0.921 | 26 M |
| Inception_v3 | 0.779 | 0.937 | 24 M |
| MobileNetV2 | 0.713 | 0.901 | 3.5 M |
| Xception | 0.790 | 0.945 | 23 M |
| EfficientNet-B0 | 0.763 | 0.932 | 5.3 M |
| EfficientNet-B2 | 0.798 | 0.949 | 9.2 M |
| EfficientNet-B3 | 0.811 | 0.955 | 12 M |

Table 1 lists all the deep learning architectures used in the project. It shows the number of parameters and performance results (Top-1 and Top-5 validation accuracy) on the popular ImageNet-1K dataset. We picked these networks to show the evolution in the architectures in chronological order. Some key features of these models are:

1. **VGG16:** It is characterized by its simplicity, using only **3×3 convolutional layers stacked on top of each other** in increasing depth. Reducing volume size is handled by max pooling. However, it has a colossal **138M parameters** and thus takes much time to train.[7]
2. **ResNet50:** The paper on residual networks has been one of the most influential papers in the Deep learning community. It introduced Residual mappings (or **skip connections**), which allowed for training much deeper networks than what was possible at the time due to issues such as the **vanishing gradient problem and the degradation problem**.[8]
3. **Inception V3:** The **inception module** introduced by Google was focused on building wider and deeper networks while keeping the computational budget constant using **1×1 convolutions for dimensionality reduction**.[9]
4. **MobileNetV2:** It was proposed as **an efficient model for mobile and embedded vision applications**. It also uses depthwise separable convolutions to build light weight deep neural networks. It only has **3.5M parameters**. And yet, its performance on ImageNet is **on par with VGG16** (has 138M parameters, **≈40 times** more than MobileNetV2). This shows the great strides made in algorithmic improvements and network design over the years.[10]
5. **Xception:** It is an extension of the Inception architecture which replaces the standard Inception modules with **depthwise separable convolutions** (i.e., a depthwise convolution followed by a pointwise convolution). It provides significant performance benefits owing to the **reduction in both parameters and mult-add operations**. [11]
6. **EfficientNets:** It proposed a **novel model scaling method** that uses a simple yet highly effective compound coefficient to scale up CNNs in a more structured manner. Unlike conventional approaches that arbitrarily scale network dimensions, such as **width, depth, and resolution**, their method uniformly scales each dimension with a fixed set of **scaling coefficients**. These networks have state of the art performance on multiple benchmarks. [6]

### 3.5 Implementation Details

- **Data Generator:** This is used to circumvent the issue of being unable to load and process a massive dataset due to memory-space limitations. A data generator is used to generate batches of data in real-time and feed it to the network directly. In the data generator class implementation, we perform **real-time image augmentation and Windowing operations** as discussed earlier (i.e., obtaining the Brain, Subdural, and Soft tissue windows from the DICOM and using it as channels of the CT image).
- **Image augmentation:** We perform **horizontal flipping** with a probability of 0.25, **vertical flipping** with a lower probability of 0.10 and **cropping** with a probability of 0.25. It was decided not to use other augmentations such as color jitter, affine transformation, rotations, etc. due to the sheer size of the training dataset.
- **Network modifications:** The **convolutional base** of the deep learning models (section 3.4) was used as a **feature extractor**. A single dense layer of 6 units (one unit for each class) was added to the convolutional base output (after Global average pooling). **ImageNet pretrained weights** were used for all models. Thus, transfer learning is performed on our medical image dataset using Imagenet weights. Due to the significant difference in the image domain between the Imagenet classes and head CT images, it was decided to **fine-tune all the layers** (i.e., all layer parameters were trainable and amenable to gradient updates).
- **Loss function: Binary cross-entropy loss** was used. As it is a multi-label classification problem, a **sigmoid activation function is used for the output layer**. If it were a multi-class and not multi-label classification, a softmax activation with categorical cross-entropy loss would have been used. However, in our case, a single image can have more than one cerebral hemorrhage. Thus, a sigmoid activation with a BCE was used.
- **Metrics:** We track **two metrics: the accuracy and the AUC score**. Due to substantial class imbalance, accuracy can be misleading. As even for a dummy classifier (i.e., a classifier that always predicts the majority class, in our case, it is 0), the accuracy is high. Thus, we also keep track of the Area under the ROC curve, which considers the Precision-Recall tradeoff. It essentially indicates how well the probabilities from the positive classes are separated from the negative classes.
- **Optimizer: Adam** was used. The main difference between Adam and Vanilla Gradient descent (GD) is that Adam optimizer tries to determine the **adaptive** learning rate. Whereas, GD assumes a fixed learning rate. Adaptive methods like Adam, estimate the change in gradient (i.e., the hessian) by keeping track of the past gradients (first moment) and past squared gradients (second moment). This leads to better updates and faster convergence while keeping the computational time similar to first-order methods (like SGD).
- **Epochs:** Each model was trained for only **10 epochs**. We aimed to compare the performance of different models and not to maximize the validation accuracy for each model.
- **Learning Rate:** We kept a learning rate of **0.0001**. A low learning rate was picked because high learning rates increase the risk of losing previous knowledge of the pretrained parameters. As high learning rates lead to more extensive gradient updates. Thus, for fine-tuning the network, we used a lower learning rate.
- **Batch size:** We kept a fixed **batch size of 32**. Much research has shown that a mini-batch size of 2-32 yields more stable and generalizable results on multiple benchmarks than large mini-batches.[12]

- **Callbacks:** We only used a **ModelCheckpoint** to save the model parameters having the best validation accuracy. **No learning rate scheduler or early stopping** was used.

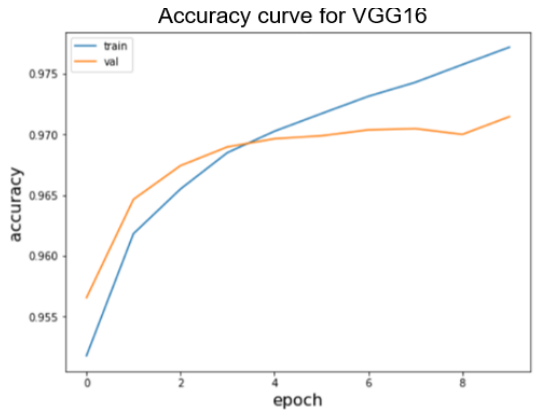
# 4 Results

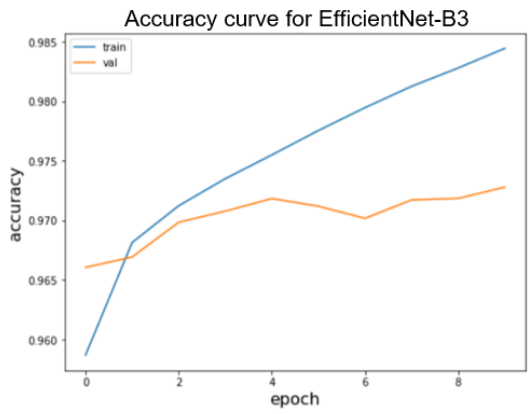
Table 2: Performance results on RSNA Intracranial Hemorrhage Detection.[5, 6]

| Model | Top-1 Acc. | Top AUC score | #Params |
|---|---|---|---|
| VGG16 | 0.9704 | 0.9619 | 138 M |
| ResNet50 | 0.9713 | 0.9730 | 26 M |
| Inception_v3 | 0.9727 | 0.9761 | 24 M |
| MobileNetV2 | 0.9700 | 0.9714 | 3.5 M |
| Xception | 0.9728 | 0.9769 | 23 M |
| EfficientNet-B0 | 0.9711 | 0.9733 | 5.3 M |
| EfficientNet-B2 | 0.9728 | 0.9756 | 9.2 M |
| EfficientNet-B3 | **0.9729** | **0.9790** | 12 M |

Table 2 lists all the deep learning architectures used in the project, their Top-1 Accuracies, AUC scores, and the number of parameters. Unfortunately, the difference in results is not as significant as what was expected. However, some trends can still be observed. These are:

1. EfficientNet-B3 was the best performing model in terms of accuracy and AUC score.
2. From the family of EfficientNets, we observe that with an increase in model capacity (number of parameters) and scale (depth, width, and resolution), there is an improvement in both accuracy and AUC score. In simpler terms, the metrics improve from B0 $\rightarrow$ B2 $\rightarrow$ B3.
3. VGG16 (the oldest network) performed the worst, even though it has the highest model capacity of 138M parameters.
4. MobileNet outperformed VGG16 by almost 1% in AUC score with 40 times fewer parameters.
5. ResNet50 and EfficientNet-B0 show similar performance. EfficientNet-B0 has five times fewer parameters than ResNet50. These two networks have been compared in [6] and Table 1.
6. Xception model performed on par with EfficientNet-B2 and B3. However, it has more parameters than both the models combined.
7. Xception being an extension of Inception_V3, shows better empirical performance than it.



# 5 Conclusion

## 5.1 Objectives Achieved

1. We built an **end-to-end deep learning pipeline** consisting of:
   - Data handling, cleaning, and pre-processing.
   - Model implementation using **Transfer learning**.
   - Model evaluation (using accuracy and AUC score)
   - Comparative analysis of eight deep learning models.
2. **Windowing**, a technique used by radiologists while analyzing CT scans, was performed in the pre-processing stage to obtain the channels of the image. Thus, we successfully were able to **extend the domain knowledge** in medical image analysis to our deep learning solution.
3. We obtained the **accuracy and AUC scores for eight deep learning models**. We provided background on the key features introduced in each model and attempted on performing a comparative empirical study of these models on a medical image analysis dataset.
4. We empirically showed that the performance of the models on the medical image dataset exhibits similar trends to the ImageNet benchmark.

## 5.2 Challenges Faced

1. Dealing with big data and data handling was a huge issue. Our computational workflow was very inefficient for the problem at hand. This cost us experimentation time.
2. The final accuracies and AUC scores of the vast range of models were underwhelming. We had expected to see a wide range of results from the models. And we hoped to exhibit the difference in the empirical performance of the older networks compared to the newer ones.
3. The colossal training duration did not allow us to try different loss functions (such as weighted cross-entropy), upsampling, and downsampling techniques. Also, the models were run only for a few numbers of epochs.
4. We also could not perform any Ablation studies (due to training time) to see the difference between introducing/removing certain parts of the pipeline to understand the network's behavior better.
5. In hindsight, the dataset we had chosen was the source of all our challenges.

# 6 Code

The code for this project is available in the link below. It contains the README markdown file, the code, and the best model weights (for each of the eight models). The link to the dataset (hosted on kaggle) is provided in the README file. Due to the size of the dataset, it is not uploaded on the drive. Screenshots of the training of each model (10 epochs) are provided in the *training runs* folder.

Drive Link: https://drive.google.com/open?id=1QZNXEgBdR6OAEm3IYV1ecEibJT2e4YLc

# References


[1] https://www.kaggle.com/c/rsna-intracranial-hemorrhage-detection

[2] https://www.kaggle.com/c/rsna-intracranial-hemorrhage-detection/overview/hemorrhage-types

[3] https://radiopaedia.org/articles/windowing-ct?lang=gb

[4] https://radiopaedia.org/articles/ct-head-an-approach?lang=gb

[5] https://keras.io/applications/

[6] Mingxing Tan, Quoc V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019.

[7] A. Z. Karen Simonyan, "Very deep convolutional networks for large-scale image recognition," 2014.

[8] Kaiming He Et al., "Deep residual learning for image recognition," 2015.

[9] Christian Szegedy Et al., "Going deeper with convolutions," 2014.

[10] Andrew G. Howard Et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[11] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2016.

[12] C. L. Dominic Masters, "Revisiting small batch training for deep neural networks," 2018.