

# Detecção de Hemorragia Intracraniana

...

Nathan Carlos  
Thiago Emanuel

# SUMÁRIO

- Motivação;
- Objetivo;
- Hemorragia intracraniana
  - Tipos;
- Bibliotecas importantes;
- EfficientNets
  - 2D Depthwise convolution
- Metodologia;
- Resultados;
- Conclusão;
- Trabalhos futuros
- Referências

# MOTIVAÇÃO

- A hemorragia intracraniana necessita de um tratamento médico rápido e intensivo;
- A detecção humana é muito difícil pelo fato de conseguirmos observar poucas escalas de cinza;
- Identificar a localização e o tipo de qualquer hemorragia é uma etapa crítica no tratamento do paciente;

# OBJETIVO

- Desenvolver um modelo para detectar hemorragia intracraniana e os seus subtipos;
- Dataset fornecido pela Sociedade Radiológica da América do Norte (RSNA).

# HEMORRAGIA INTRACRANIANA

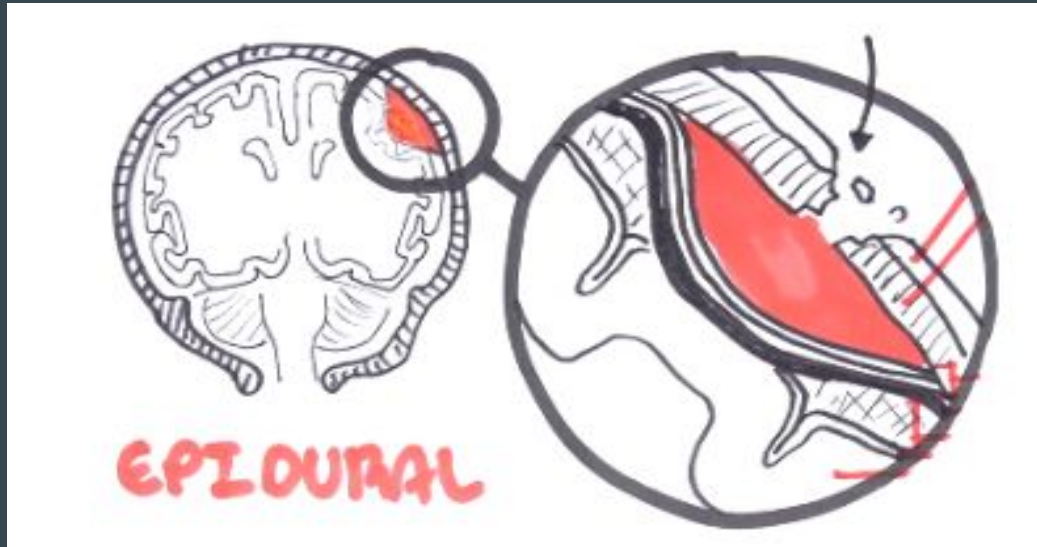
- É o sangramento dentro do crânio;
- O acúmulo de sangue no crânio pode levar a aumentos da pressão intracraniana , que podem esmagar tecidos cerebrais delicados ou limitar o suprimento sanguíneo;
- Pode ser causado por trauma, acidente vascular cerebral, aneurisma, malformações vasculares, pressão alta, drogas ilícitas e distúrbios de coagulação do sangue

# TIPOS

- São separados em dois grupos: extra e intra
  - Extra:
    - epidural
    - subdural
    - Subaracnóide.
  - Intra:
    - Intraparenquimal
    - intraventricular.

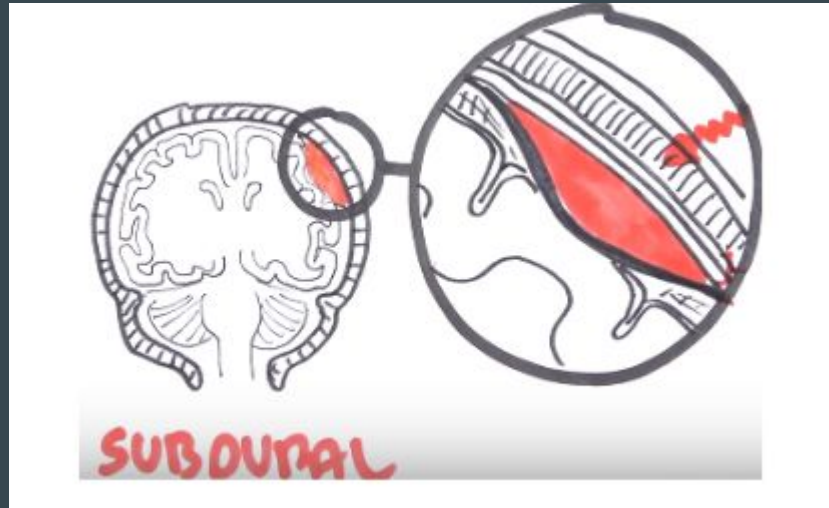
## Extra: Epidural

- O sangue fica entre o crânio e a camada mais externa do tecido do cérebro.



## Extra: Subdural

- O sangue fica entre a camada externa e média do tecido do cérebro.





# Extra: Subaracnóide

- Quando acontece uma ruptura de uma artéria chamada aneurisma fazendo com que o sangue ocupe o espaço subaracnóide.



# Intra: Intraparenquimal

- A hemorragia acontece dentro do cérebro.



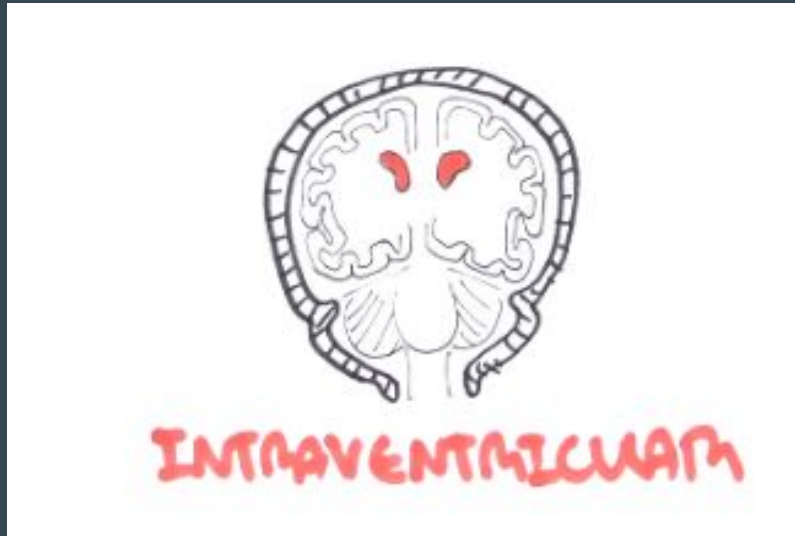
INTRACEREBRAL



INTRACEREBRAL

# Intra: Intraventricular

- Acontece dentro do ventrículo



# Sobre os dados

- 250 GB de imagens
- Existem duas tabelas: “submission.csv” e “train.csv”. Onde a primeira contém os formatos corretos de cada imagem e a segunda o treino.
- A divisão por tipo de imagem é feita da seguinte forma:
  - ID\_(tipo da hemorragia).dcm, (probabilidade)

```
ID,Label
ID_12cad6af_epidural,0
ID_12cad6af_intraparenchymal,0
ID_12cad6af_intraventricular,0
ID_12cad6af_subarachnoid,0
ID_12cad6af_subdural,0
ID_12cad6af_any,0
ID_38fd7baa0_epidural,0
ID_38fd7baa0_intraparenchymal,0
ID_38fd7baa0_intraventricular,0
ID_38fd7baa0_subarachnoid,0
ID_38fd7baa0_subdural,0
ID_38fd7baa0_any,0
ID_6c5d82413_epidural,0
ID_6c5d82413_intraparenchymal,0
ID_6c5d82413_intraventricular,0
ID_6c5d82413_subarachnoid,0
ID_6c5d82413_subdural,0
ID_6c5d82413_any,0
ID_aec8e68b3_epidural,0
ID_aec8e68b3_intraparenchymal,0
ID_aec8e68b3_intraventricular,0
ID_aec8e68b3_subarachnoid,1
ID_aec8e68b3_subdural,0
ID_aec8e68b3_any,1
ID_4d9209c7c_epidural,0
ID_4d9209c7c_intraparenchymal,0
ID_4d9209c7c_intraventricular,0
```

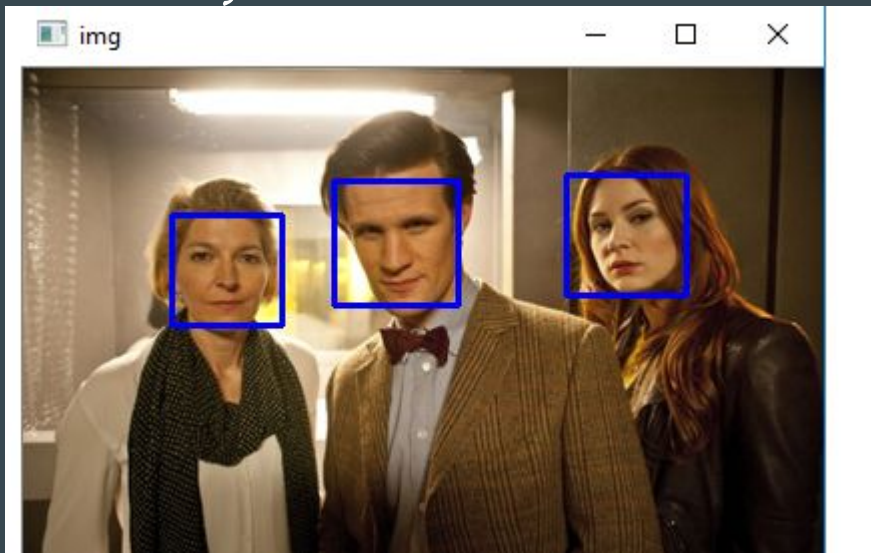
- Uma imagem pode ou não ser uma hemorragia como também pode ser várias. Caso não seja nenhuma, a coluna label precisa ser toda 0.
- Caso tenha alguma hemorragia, a label da mesma precisa ser preenchida com 1, e além disso, a imagem com o tipo “any” também precisa ser 1. Pois, também pertence a classe de qualquer hemorragia.

# Bibliotecas importantes usadas

- cv2
- imgaug
- pydicom
- efficientNet
















# cv2

- É usada para diversos tipos de análise em imagens e vídeos, como detecção, tracking e reconhecimento facial, edição de fotos e vídeos, detecção e análise de textos, etc.



# imgaug








- Converte um conjunto de imagens de entrada em um novo conjunto muito maior de imagens ligeiramente alteradas.

	Image	Heatmaps	Seg. Maps	Keypoints	Bounding Boxes, Polygons
<i>Original Input</i>					
Gauss. Noise + Contrast + Sharpen					
Affine					



# pydicom

- pydicom é um pacote python puro para trabalhar com arquivos DICOM.

 ID_000012eaf.dcm	30/10/2019 18:09	Arquivo DCM	513 KB
 ID_000039fa0.dcm	30/10/2019 16:00	Arquivo DCM	513 KB
 ID_00005679d.dcm	30/10/2019 15:13	Arquivo DCM	513 KB
 ID_00008ce3c.dcm	30/10/2019 15:38	Arquivo DCM	513 KB
 ID_0000950d7.dcm	30/10/2019 15:52	Arquivo DCM	513 KB
 ID_0000aee4b.dcm	30/10/2019 17:12	Arquivo DCM	513 KB
 ID_0000ca2f6.dcm	30/10/2019 18:09	Arquivo DCM	513 KB

# Modelo EfficientNet

- As EfficientNets são uma família de modelos de classificação de imagens, que atingem precisão de última geração, sendo uma ordem de magnitude menor e mais rápida que os modelos anteriores.
- Por ser um modelo de aprendizagem para base de dados profundas, como a que utilizamos foi por esse motivo que resolvemos utilizá-la.

# Modelo EfficientNet

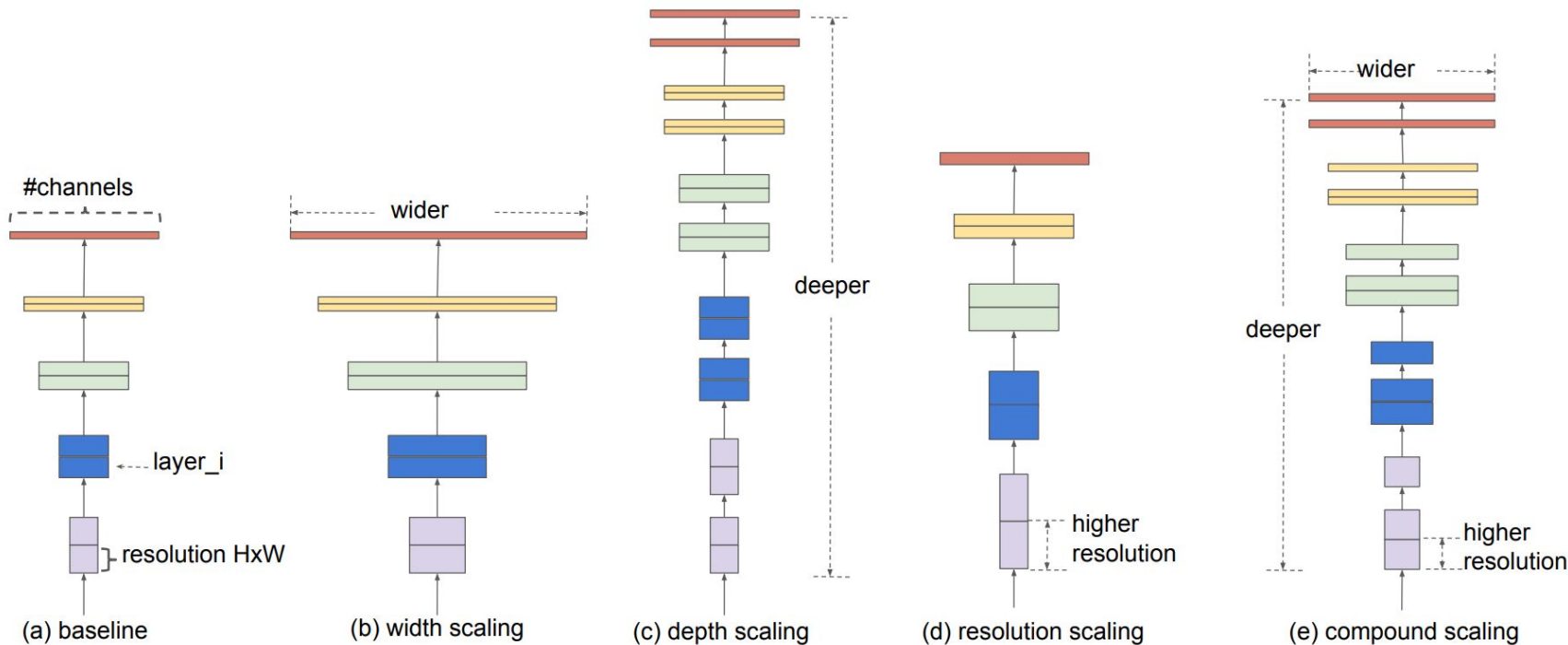
- Esse modelo Procura dimensionar a rede de maneira mais eficiente, equilibrando cuidadosamente a profundidade, largura e resolução (As 3 dimensões de uma rede neural convolucional) o que leva a um melhor desempenho;
- **Profundidade:** Número de camadas da rede;
- **Largura:** Número de canais em uma camada de convolução;
- **Resolução:** Resolução da imagem que está sendo passada para a rede.

# Modelo EfficientNet

- **Arquiteturas:** A arquitetura base da EfficientNet é a B0, é construída em torno da *2D Depthwise convolution* (Convolução 2D profunda).
- Seguindo a estratégia de aumentar a profundidade e largura da rede, de modo a melhorar os resultados da rede, pode-se escolher arquiteturas maiores: B1, B2, B3, etc.
- Cada número representa um tamanho de rede e o poder de processamento praticamente dobra a cada incremento.
- Neste trabalho, foi utilizado a EfficientNet com arquitetura B2, por necessitarmos de um modelo robusto mas que não tivesse um custo computacional tão elevado como as de arquiteturas superiores..

# EfficientNet: Comparação de diferentes métodos de dimensionamento

## EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

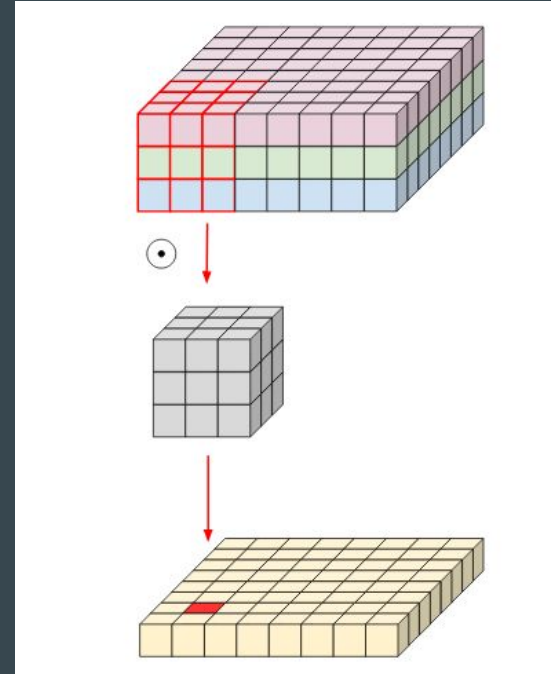


# 2D Depthwise convolution

- Base da EfficientNet
- É muito similar a um 2D Convolution block (bloco de convolução 2D)
  - O bloco de convolução 2D representa uma camada que pode ser usada para detectar recursos espaciais em uma imagem, trabalhando diretamente nos dados da imagem ou na saída dos blocos de convolução anteriores.
- Os bloco de convolução 2D Profundos produzem mais canais de saída com a mesma quantidade de cálculo de um bloco de convolução 2D, tornando-o mais eficiente em algumas situações.

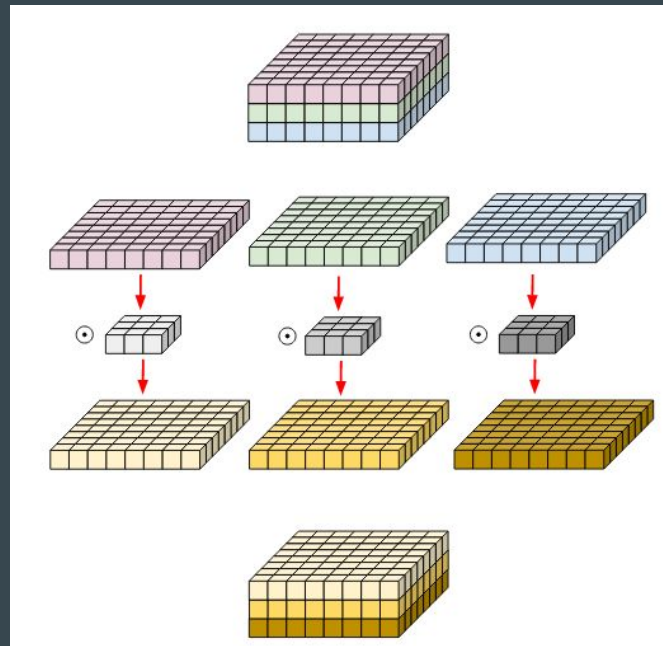
# 2D Depthwise convolution

- Convolução normal:

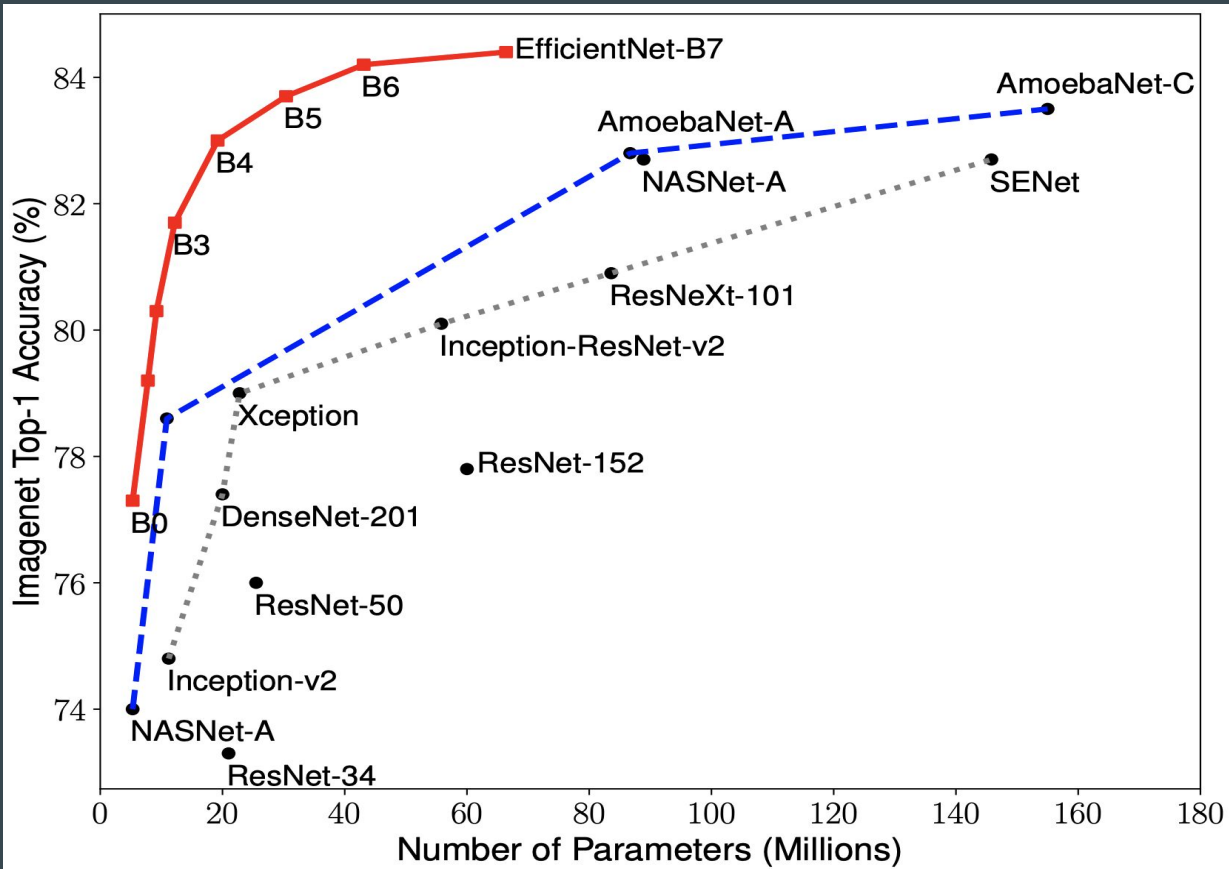


# 2D Depthwise convolution

- Convolução profunda:
  - Os filtros e a imagem são divididos em três canais diferentes, convoluídos separadamente e empilhados posteriormente.







- Gráfico de algumas aprendizagens em relação à acurácia e número de parâmetro, enfatizando as efficientnets.

# Metodologia

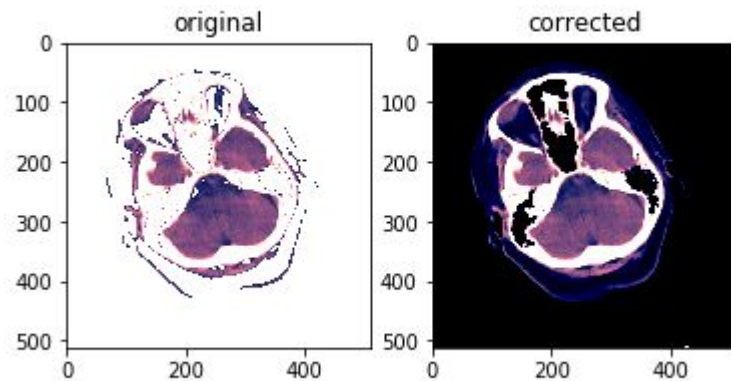
- Pré-processamento:
  - Corrigir as imagens .dcm
  - Windowing
  - Augmentation
- Data generation
- Teste
- Aplicar modelo EfficientNetB2

# Correção das imagens

- Os valores da imagem correspondem às unidades de Hounsfield (HU). Mas os valores armazenados no Dicom não são unidades de Hounsfield, mas uma versão em escala. Para extrair as unidades de Hounsfield, precisamos aplicar uma transformação linear, que pode ser deduzida das tags Dicom.

```
def correct_dcm(dcm):  
    x = dcm.pixel_array + 1000  
    px_mode = 4096  
    x[x>=px_mode] = x[x>=px_mode] - px_mode  
    dcm.PixelData = x.tobytes()  
    dcm.RescaleIntercept = -1000
```

Sem correção / com correção



# Windowing

- Foi feito também o windowing para colocar as imagens em um tamanho predefinido (janela) a fim de melhor precisão.

```
def window_image(dcm, window_center, window_width):  
    if (dcm.BitsStored == 12) and (dcm.PixelRepresentation == 0) and (int(dcm.RescaleIntercept) > -100):  
        correct_dcm(dcm)  
    img = dcm.pixel_array * dcm.RescaleSlope + dcm.RescaleIntercept  
  
    # Resize  
    img = cv2.resize(img, SHAPE[:2], interpolation = cv2.INTER_LINEAR)  
  
    img_min = window_center - window_width // 2  
    img_max = window_center + window_width // 2  
    img = np.clip(img, img_min, img_max)  
    return img
```



# Data generation

- Foi feita a predição com 16 batch no teste e treino. Ou seja, o data era dividido em 16 partes para ser chamado na hora do treino e teste.

```
class TrainDataGenerator(keras.utils.Sequence):  
    def __init__(self, dataset, labels, batch_size = 16, img_size  
= SHAPE, img_dir = TRAIN_IMAGES_DIR, augment = False, *args, **kwa  
rgs):
```

```
class TestDataGenerator(keras.utils.Sequence):  
    def __init__(self, dataset, labels, batch_size = 16, img_size  
= SHAPE, img_dir = TEST_IMAGES_DIR, *args, **kwargs):
```

# Teste

- O treino e teste foi feito por estratificação, ou seja, fazendo com que as classes não fiquem desbalanceadas.

```
# Multi Label Stratified Split stuff...
msss = MultilabelStratifiedShuffleSplit(n_splits = 10, test_size = TEST_SIZE, random_state = SEED)
X = train_df.index
Y = train_df.Label.values

# Get train and test index
msss_splits = next(msss.split(X, Y))
train_idx = msss_splits[0]
valid_idx = msss_splits[1]
```



# Teste

- A predição do teste foi feito da seguinte forma:

```
def predictions(test_df, model):  
    test_preds = model.predict_generator(TestDataGenerator(test_df, None, 8, SHAPE, TEST_IMAGES_DIR), verbose  
    return test_preds[:test_df.iloc[range(test_df.shape[0])].shape[0]]
```

```
if epoch >= 2:  
    preds = predictions(test_df, model)  
    submission_predictions.append(preds)
```

- A partir de uma certa época, é feito o teste.

# Validação

O treino e a validação foi feita pegando os índices dos testes que foram feitos da estratificação.

```
# Get train and test index  
msss_splits = next(msss.split(X, Y))  
train_idx = msss_splits[0]  
valid_idx = msss_splits[1]
```

```
# Create Data Generators for Train and Valid  
data_generator_train = TrainDataGenerator(train_df.iloc[train_idx],  
                                           train_df.iloc[train_idx],  
                                           TRAIN_BATCH_SIZE,  
                                           SHAPE,  
                                           augment = True)  
data_generator_val = TrainDataGenerator(train_df.iloc[valid_idx],  
                                         train_df.iloc[valid_idx],  
                                         VALID_BATCH_SIZE,  
                                         SHAPE,  
                                         augment = False)
```

```
# Create Model
```

# Criando o modelo

```
def create_model():  
    K.clear_session()  
  
    base_model = efn.EfficientNetB2(weights = 'imagenet', include  
_top = False, pooling = 'avg', input_shape = SHAPE)  
    x = base_model.output  
    x = Dropout(0.15)(x)  
    y_pred = Dense(6, activation = 'sigmoid')(x)  
  
    return Model(inputs = base_model.input, outputs = y_pred)
```

- O modelo escolhido foi a EfficientNetB2. Porém, poderia ser escolhido qualquer outra, pois o que muda são só as profundidades em que é tratado (parâmetros).

# Treinando modelo

```
# Train Model
model.fit_generator(generator = data_generator_train,
                    validation_data = data_generator_val,
                    steps_per_epoch = TRAIN_STEPS,
                    epochs = 1,
                    callbacks = [ModelCheckpointFull('model.h
5')],

                    verbose = 1)
```

# Resultados

- Por esse problema ser um desafio da Kaggle, é necessário salvar as predições do conjunto de teste em um arquivo csv e submeter para que a Kaggle calcule a acurácia (score).
- Foram feitas vários testes submetido no kaggle, obtendo os scores que segue

# Resultados

- dropout 0.5 e função de ativação sigmoid e obtido esses resultados.
- Score: 0.28597

	ID	Label
0	ID_000000e27_any	0.056319
1	ID_000000e27_epidural	0.003606
2	ID_000000e27_intraparenchymal	0.004152
3	ID_000000e27_intraventricular	0.000606
4	ID_000000e27_subarachnoid	0.021173
5	ID_000000e27_subdural	0.037575
6	ID_000009146_any	0.000230
7	ID_000009146_epidural	0.000024
8	ID_000009146_intraparenchymal	0.000108
9	ID_000009146_intraventricular	0.000050

**Obs:** O processo de treinamento, por questão de pouco desempenho de máquina, foi realizado com apenas 3 épocas.

# Resultados

	ID	Label
0	ID_000000e27_any	0.077848
1	ID_000000e27_epidural	0.005943
2	ID_000000e27_intraparenchymal	0.005107
3	ID_000000e27_intraventricular	0.000446
4	ID_000000e27_subarachnoid	0.035602
5	ID_000000e27_subdural	0.049312
6	ID_000009146_any	0.000242
7	ID_000009146_epidural	0.000148
8	ID_000009146_intraparenchymal	0.000038
9	ID_000009146_intraventricular	0.000013

Segue alguns resultados de alguns outros testes feitos mudando alguns parâmetros

- Com dropout 0,15 e função sigmoid
- Score: 0.40740

# Resultados

	ID	Label
0	ID_000000e27_any	0.239013
1	ID_000000e27_epidural	0.116627
2	ID_000000e27_intraparenchymal	0.149381
3	ID_000000e27_intraventricular	0.104596
4	ID_000000e27_subarachnoid	0.172284
5	ID_000000e27_subdural	0.218099
6	ID_000009146_any	0.156603
7	ID_000009146_epidural	0.170908
8	ID_000009146_intraparenchymal	0.173923
9	ID_000009146_intraventricular	0.171047

- Com dropout 0,5 e função softmax
- Score: 0.53651



# Resultados

	ID	Label
0	ID_000000e27_any	0.208929
1	ID_000000e27_epidural	0.129458
2	ID_000000e27_intraparenchymal	0.161463
3	ID_000000e27_intraventricular	0.118203
4	ID_000000e27_subarachnoid	0.177267
5	ID_000000e27_subdural	0.204680
6	ID_000009146_any	0.155859
7	ID_000009146_epidural	0.159233
8	ID_000009146_intraparenchymal	0.165837
9	ID_000009146_intraventricular	0.174159

- Com dropout 0,15 e função softmax
- Score: 0.46050

# Conclusão

Em questão de scores submetidos ao kaggle, a função softmax com dropout 0.5 se saiu melhor nesse problema.

Um dos possíveis motivos para que o modelo não obteve um desempenho superior é por ter sido treinado com poucas épocas. Além disso, a grande quantidade de dados e o baixo poder computacional dificultaram a realização de mais testes.

# Trabalhos Futuros

Em trabalhos futuros, deseja-se treinar o modelo utilizando mais épocas. Além de usar outras arquiteturas superiores da EfficientNet.

# Referências

<https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/pretrained-snippets/efficientnet---pretrained>

<https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/2d-depthwise-convolution>

<https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html>