PRACTICAL APPLICATIONS OF DEEP LEARNING

PROJECT:
ACUTE BRAIN INFARCTS CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS

By: SHRINIDHI CHORAGI

PROBLEM STATEMENT / TECH OVERVIEW

"To identify the Acute brain Infarct location from a given MRI image of the patient's brain"

Data Organizing → **Data Pre-processing** → **CNN Modelling** → **Model Optimizer** → **Inference**

- **Data Organizing**: Location-wise, Merging
- **Data Pre-processing**: Cropping, Resizing, Augmenting
- **CNN Modelling**: Keras
- **Model Optimizer**: IR conversion
- **Inference**: novel data prediction

DATA GATHERING

- Dataset with **50 cases** (ADC, FLAIR and DWI) & **PDF** with location of the acute infarcts
- Organized data - **"CLEANED_DATA"** folder

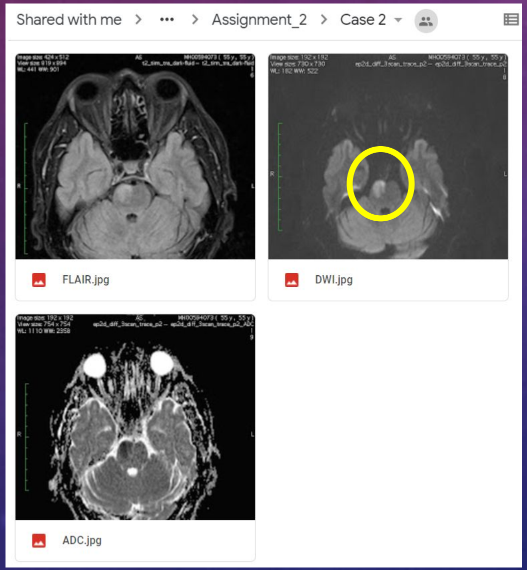
  ↳ **Subfolders (46)** - named as infarct location
- **Merging similar location classes (36) -** To reduce no. of output classes

'Lacunar infarct in right parietal lobe' + 'Right parietal lobe' = 'Right parietal lobe'

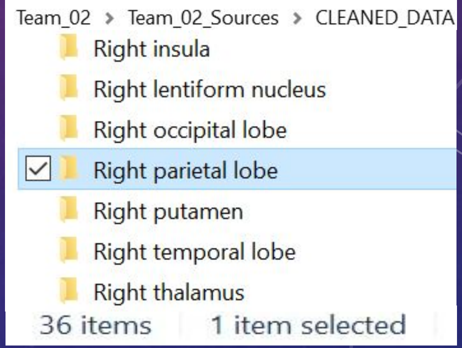
- Only **DWI (Diffusion-Weighted Imaging)** used - sensitive to **acute stroke (water movements)**

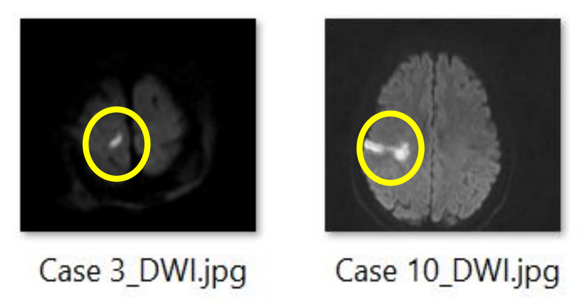
ACUTE INFARCTS

Sl.	Name	Age/Sex	Patient ID	MRI No.	Location of acute infarct
1.	P1	28/M	593583	11074	Splenium of the corpus callosum
2.	P2	55/M	594073	11075	Pontine infarct on the right
3.	P3	D2/M	709659	11079	Lacunar infarcts in the right parietal lobe
4.	P4	75/F	710152	11116	Left centrum semi ovale and right parietal lobe
5.	P5	75/M	387048	11131	Right cerebellar hemisphere
6.	P6	35/F	594751	11135	Right frontal lobe



46 classes → 36 classes





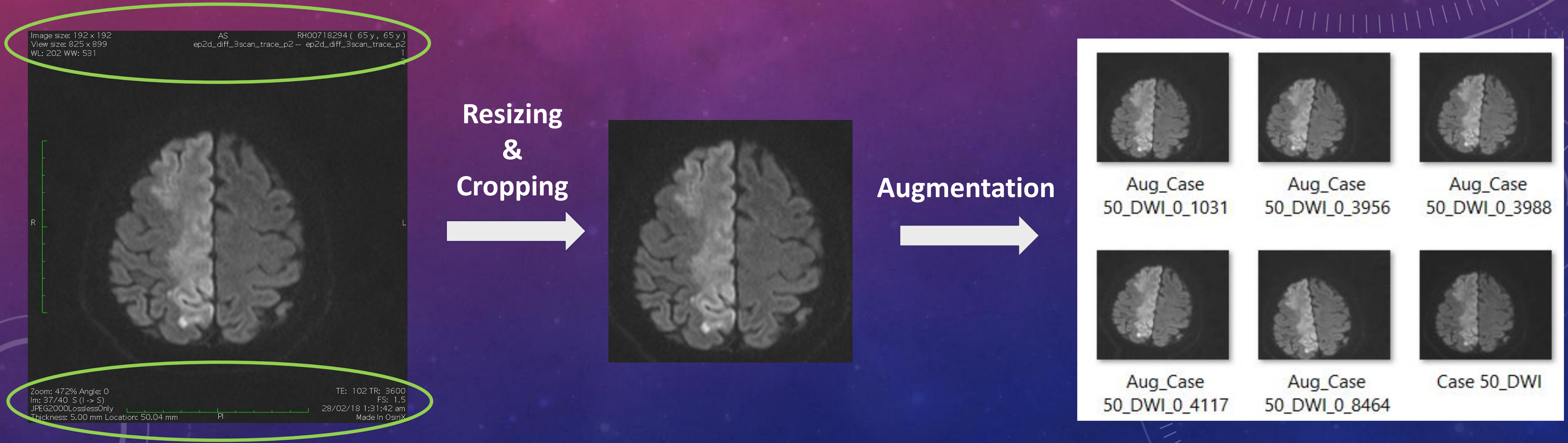
3.	P3	D2/M	709659	11079	Lacunar infarcts in the right parietal lobe
10.	P10	57/F	595192	11181	Right parietal lobe

DATA PRE-PROCESSING & AUGMENTATION

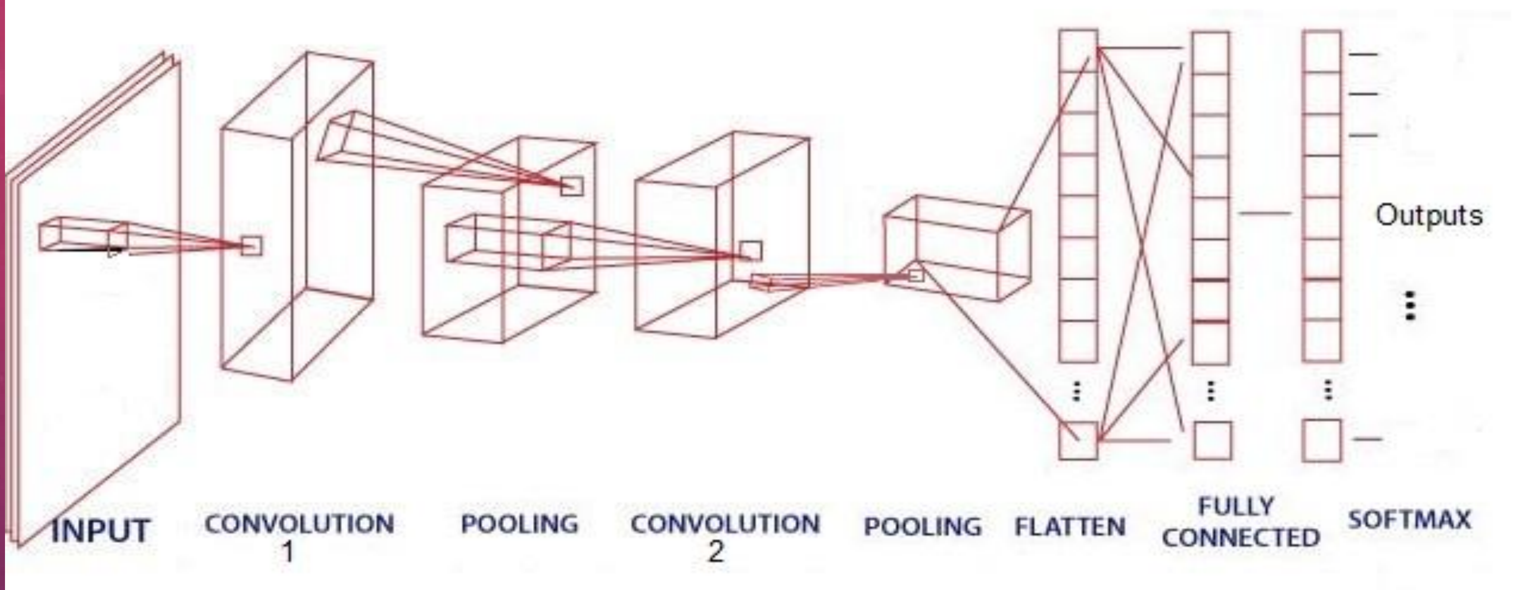
- **Resizing & Cropping -** Removing “**Textual**” components
- **Less data** thus, **Augmentation**

  Did - height, width, tilt, and shear (very less range)

  Didn’t - Vertical-flip, horizontal-flip
- 36 classes - each contains **same no. of aug** images ⟶ **unbiased model**



CNN ARCHITECTURE



Layer (type)	Output Shape	Param #
conv2d_11 (Conv2D)	(None, 124, 124, 64)	4864
max_pooling2d_11 (MaxPooling	(None, 31, 31, 64)	0
conv2d_12 (Conv2D)	(None, 27, 27, 64)	102464
max_pooling2d_12 (MaxPooling	(None, 6, 6, 64)	0
flatten_6 (Flatten)	(None, 2304)	0
dropout_6 (Dropout)	(None, 2304)	0
dense_6 (Dense)	(None, 36)	82980
activation_6 (Activation)	(None, 36)	0

- **Conv layer 1** - **64** Neurons - activation function: **ReLU**
- First **Max pooling** layer - pool size: (4,4)
- **Conv layer 2** - **64** Neurons - activation function: **ReLU**
- Second **Max pooling** layer - pool size: (2,2)
- **Flatten** layer
- **Dropout layer** (prevents neuron interdependencies and overfitting)
- **Fully Connected layer** - **36** neurons
- Output layer activation function: **Softmax**

```
Total params: 190,308
Trainable params: 190,308
Non-trainable params: 0
```

TRAIN AND TEST

- After the augmentation and grouping based on location, there are **36 classes of training images** and **10** of the repeated or closely located brain scan images are given to the **test** set.

Training and validation:

```
Number of examples is: 1115
X shape is: (1115, 128, 128, 3)
y shape is: (1115,)
```

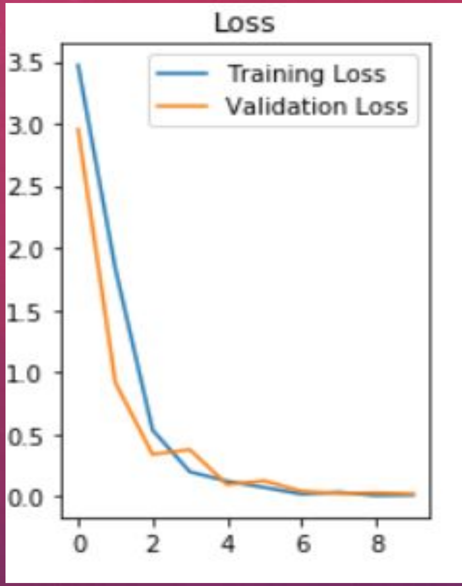
Test:

```
Number of examples is: 10
X shape is: (10, 128, 128, 3)
y shape is: (10,)
```

- In the Training Dataset we have considered a **70-30 training-validation** split

Train on: 781 samples
Validate on: 334 samples

ACCURACIES AND TWEAKS



Training loss: 0.0094
Val. loss: 0.0190

Training accuracy: 0.9978
Val. accuracy: 0.9910

Tweaks

- No. of epochs (8,**10**,12)
- Filter size (**(4,4), (5,5)**)
- Dropout (0.2,**0.5**)
- Input resize (**128, 128, 3**) values
- Degree of variation in Augmented images (10, 20, **30**)

Test set:

```
The predicted output is:
23 33 18 33 10 24 14 14 22 14
The index must be:
[23 31  6  0 10 24 25 14 33  2]
```

Numbers = **Index of class folders** when sorted alphabetically **(4/10 = 40%)**

Inference set:

```
Case 1 DWI.jpg :
Right parietal lobe
Actual: Right parietal lobe
```

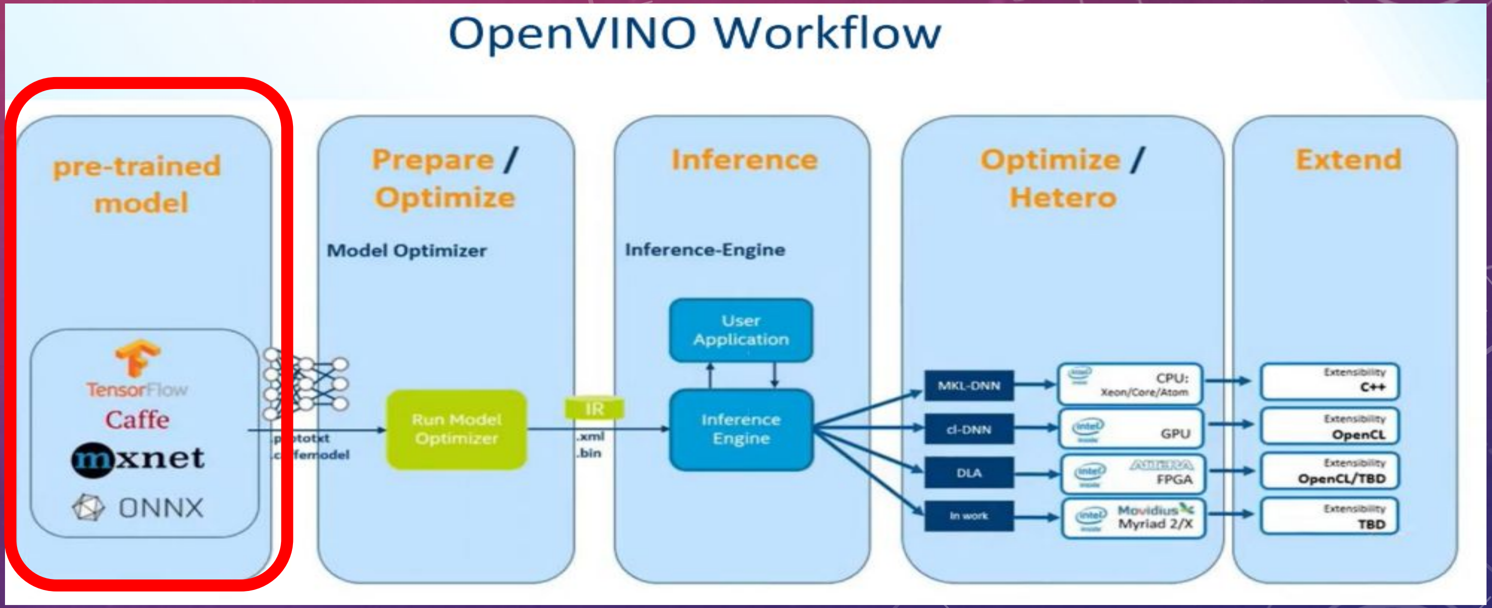
Around **6-7 out of the 36** inference cases were predicted **correctly (16.7% - 19.4%)**

INTEL MODEL OPTIMIZER

Model optimizer provides an **Intermediate Representation (IR)** of the network which can be read, loaded and inferred with the inference engine.

Optimization techniques:

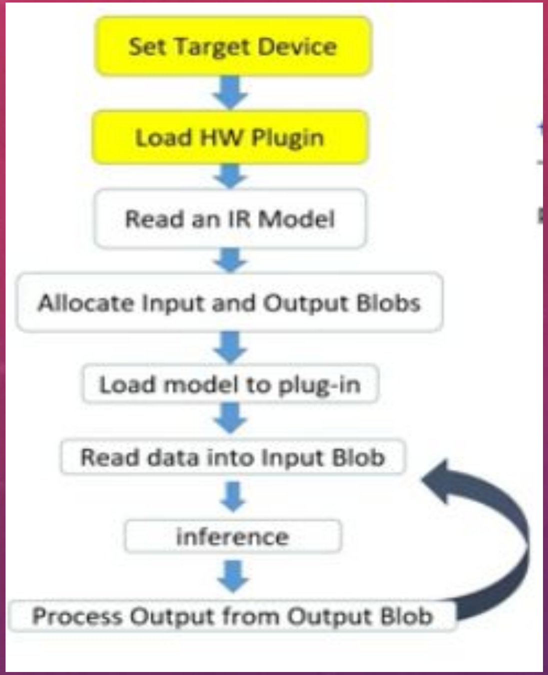
- **Quantization** - reduction of precision of weights and biases
- **Freezing** - randomly freezing layers from training to fine tune the other layers.
- **Fusion**

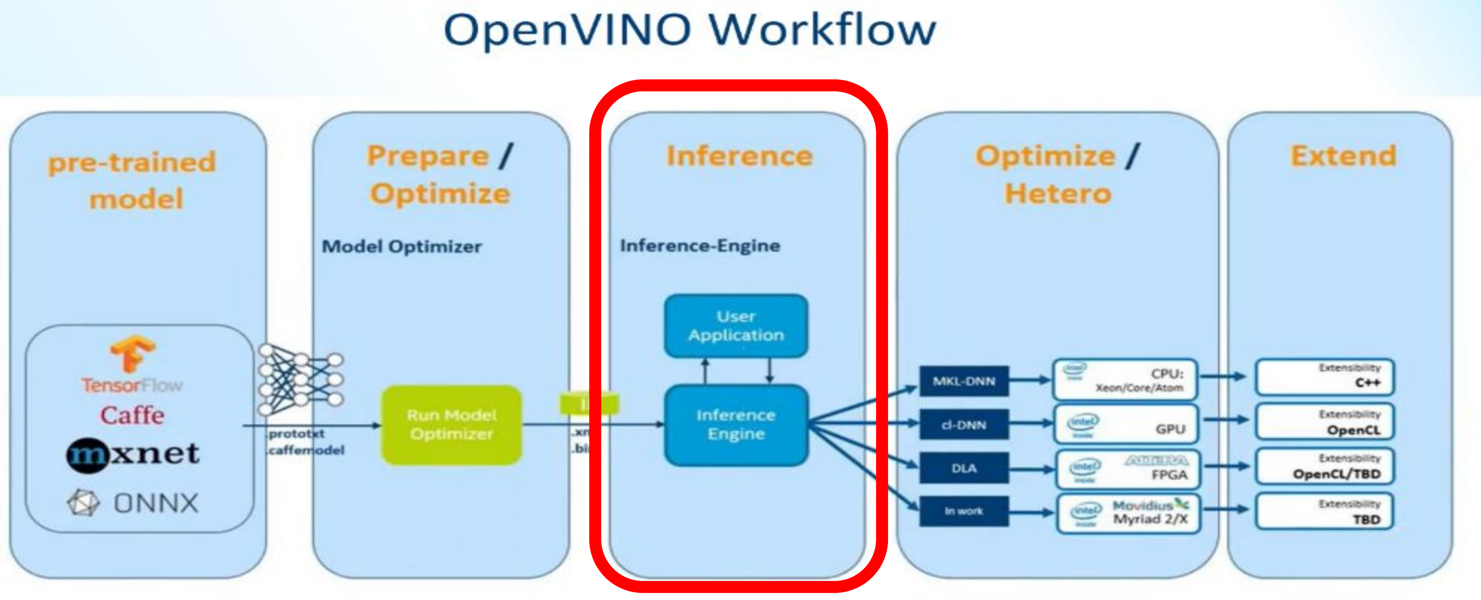


Workflow to convert the keras model to OPENVINO model and make a prediction:

1. Save the **Keras model** as a single **.h5** file
2. Load the .h5 file and **freeze the graph** to a single Tensorflow **.pb** file
3. Run the OPENVINO **mo_tf.py** script to **convert** the .pb file to IR model **(xml and bin)**
4. **Load** the model xml and bin file with **OPENVINO inference engine** and make a **prediction.**

SCRIPT TO RUN THE MODEL WITH INTEL OPENVINO





Workflow for using the **Inference Engine API**

1. **Create** an Inference Engine **core object**
2. **Read** the **Intermediate Representation** obtained from optimizer.
3. **Prepare** the **inputs** and **outputs** format
4. **Load** the network to the plugin
5. **Call** the inference API

OBSERVATIONS / CONCLUSION

- **Best accuracy** of the model : **For** epochs = 9, dropout = 0.3, batch size = 20)

Training set: 98 %	Validation set: 98 %
Test set: 40 %	Inference set: 21- 22%

- **Model performed well** with the given **small dataset.** However, **larger dataset** is required for **better prediction.**
- Epochs **over 10** cause **overfitting** and **below 8** cause **underfitting** on the current model. **Dropout** can be used to **reduce** overfitting.
- Model **predicts** some cases **semi-correctly**

```
Case 28 DWI.jpg :
Right fronto-parieto-temporo- occipital lobes
Actual: Right frontal and parietal lobes
```

- **In conclusion,**

  Intel Distribution of OpenVINO toolkit is an extremely useful framework to optimize the models and execute computer vision using deep learning on edge systems.

THANK YOU!