Master of Science in

Artificial Intelligence and Machine Learning

CS6482 Deep Reinforcement Learning

Assignment 1: Convolutional Neural Networks

Vilohit Keshava Murthy Achar

23077751

Mohit Chandrashekhar Attarde

23241977

Prof. J. J. Collins

Dept. of Computer Science & Information Systems

TABLE OF CONTENT

1. Introduction:

For learning CNN Architecture, we found various architectures from Inception, ResNet, VGG16 to EfficientNet series. We chose the VGG16 model because, while YOLO is mostly used for real-time cases and requires a larger dataset, the EfficientNet is computationally more expensive to use. We are using the CBIS DDMS (Curated Breast Imaging Subset of DDSM) dataset to classify mammography images distinguishing between benign and malignant abnormalities which has tumor (mass) type dataset and calcium deposits (calc) type dataset from (Link). The dataset is made from 1,566 participants and in DICOM format 6,671 patient cases.

2. The Dataset

The mammography images, including the MLOs (Mediolateral Obliques mammograms from the lateral side to the medial side) and CCs (Craniocaudal mammograms from top (cranial) to bottom (caudal)), consist of both malignant and benign features. Malignant are cancerous with more complex and irregular patterns whereas Benign are less cancerous/curable and have a smaller rate of growth and the patterns are well defined. Benign without callback indicates those cases where the cases were worth tracking but required no further follow-ups.

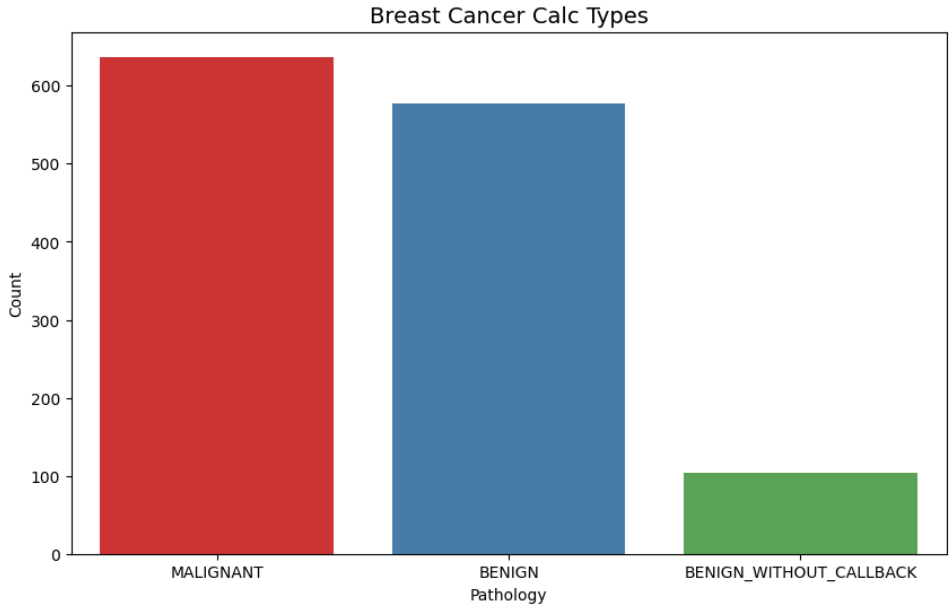

Fig 1: Number of Benign, Benign without callback and Malignant

There are two datasets, mass-type, and calc-type. Both consist of breast density, assessment, pathology type, abnormality type and image path. Fig 1 shows the classes have almost similar sample size, so it is a **balanced dataset with less bias**.

3. Preprocessing

We have DICOM dataset which has files and image path of mammograms, along with description of images as cropped, full mammograms and Region of Interest mask. We create dictionaries and modify the image path in mass and calc dataset for accessibility. [N1] As VGG16 needs an input of 224x224, we resize the images from mass and calc set from 426 to 224. Later, we replace the pathology class names with numerical labels and calculate the number of classes. [N2]

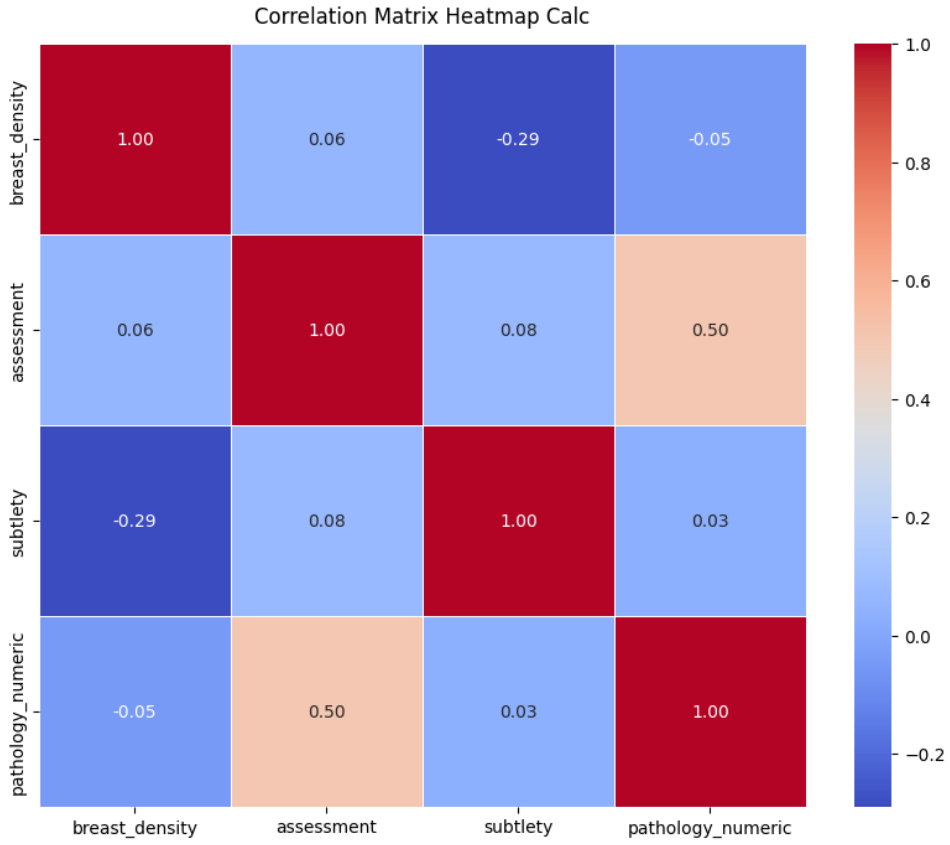

Fig 2: Correlation between breast density, assessment, sublety and pathology numeric

From the correlation matrix, breast density has a very weak positive correlation with assessment and a weak negative correlation with subtlety. It shows a very weak negative correlation with pathology numeric, indicating that breast density has little to no direct relation with the malignancy of a mass.

4. Methodology

VGG16 (Visual Geometry Group – 16 layers) was proposed by Karen Simonyan and Andrew Zisserman in their paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" on 4th Sept 2014 at ICLR 2015. [2] The model achieved 92.7% top-5 test accuracy in ImageNet. VGG16 is used in object detection and classification and is easy to use with transfer learning. Fig 3 represents overall architecture which consists of 16 learnable layers which is two setups of 2 X Conv2D and MaxPool and three setups of 3 X Conv2D and MaxPool.

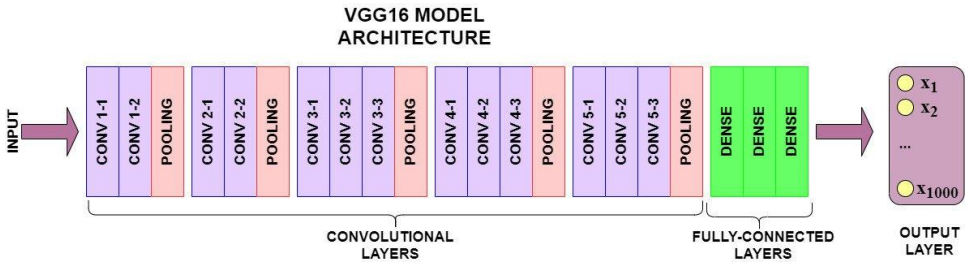

Fig 3: Basic architecture of VGG16

The input size is 224 X 224 X 3, with the Conv-1 layer having 64 filters, the Conv-2 layer having 128 filters, the Conv-3 layer having 256 filters, and both the Conv-4 and Conv-5 layers having 512 filters each. [3] The next three fully connected layers of Dense have the SoftMax layer though we have implemented sigmoid for testing results. Fig 4 shows the filter size breakdown of VGG16 model.

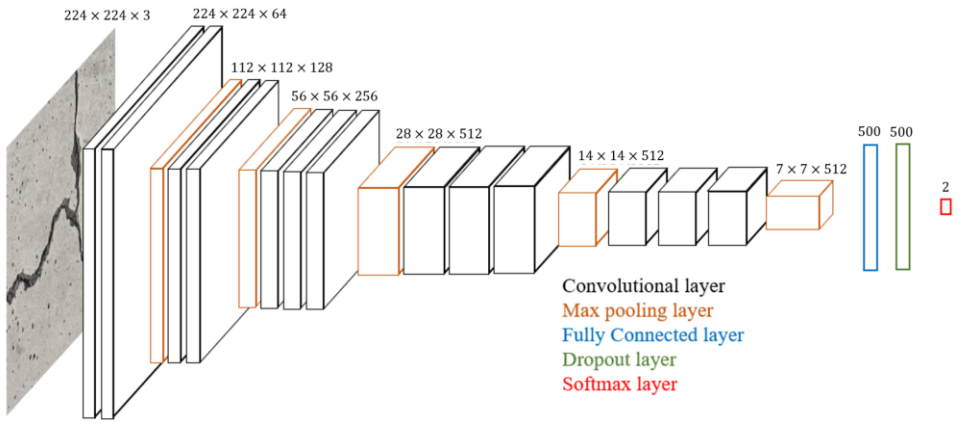

Fig 4: Visual representation of the VGG16

4.1 VGG16 with Binary Cross Entropy and Adam Optimizer

We chose to use pre-trained ImageNet weights model from Keras library without GPU excluding the top classification layer. The last layers in the pre-trained model use ReLU with L2 regularizer for complexity control on which we apply a sigmoid activation function suitable in binary classification. For optimization, we selected Adam optimizer which is a combination of AdaGrad and RMSProp. The loss function used in these models is Binary Cross-Entropy to measure the difference between the predicted and actual datapoints. The models ran for 40 epochs without early stopping to compare results of all our models together. As we have two datasets, "Model" represents the model training on mass dataset and "Model1" represents the model training on calc dataset. All models took 70 mins each on average for training. [N3]

4.2 K-fold Cross Validation

To validate, we partition the dataset into five distinct folds, to ensure that each fold served as a validation set once while the remaining folds collectively formed the training set with the same Adam optimizer with learning rate of 0.0001 and Binary Cross-Entropy loss function hyperparameters. [N4]

4.3 VGG16 with Categorical Cross Entropy and RMSProp Optimizer

Before using Categorical Cross Entropy as Loss function, we encoded the dataset with one-hot encoding. The choice of categorical cross-entropy aligns with the multi-class nature of our target. Here, we slightly modify the augmented image data generator by reducing rotation, height, width and channel shift range, shear and zoom range. We immobilized the training capability aiming to fine-tune the model; hence we used ELU activation function and RMSProp optimizer favoring a low learning rate to ensure gradual learning without overshooting minima. The final dense layer with 10 units has a softmax activation function for multi-class classification purposes. [N5]

4.4 VGG16 with Batch Normalization and 50 percent Dropout

From our results of last models, we choose to reuse our VGG16 model with Binary Cross Entropy and Adam Optimizer setup with changes in the final layers to reduce overfitting and better generalization. We added one layer of batch normalization before 128 units of dense layer which later is followed by 50 percent dropout layer to randomly omit a subset of features during training. The final layer uses a sigmoid function to output predictions for binary classification. [N6]

5. Results

In the mass dataset, our best performance came from the VGG16 setup, achieving a test accuracy of 73%. To address this performance, we experimented with various hyperparameters. The ELU Softmax model is underfitting which is evident by high training loss and low accuracy, it even struggled to generalize with 50% test accuracy which is prone to random guessing. Conversely, the Batch Normalization model showed high training accuracy being a good initial fit. However, the disparity with lower test accuracy points towards overfitting. In calc dataset, a similar pattern can be seen for all the models, which implies it has learned noise and struggling to capture underlying patterns.

Dataset Type	Model	Training Loss	Training Accuracy	E_in	Test Loss	Test Accuracy	E_out	Model Result
Mass Type	CNN Output Predictions - VGG16 - ReLU Sigmoid	0.0875	0.9963	0.0037	1.2247	0.7324	0.2676	Overfitting
Mass Type	K-fold Validation	0.4311	0.9278	0.0722	0.0828	1.0000	0.0000	Underfitting (Error)
Mass Type	Varying hyperparameters ELU Softmax	2.0336	0.5605	0.4395	2.0766	0.5000	0.5000	Underfitting
Mass Type	BatchNormalization Dropout	0.8342	0.9403	0.0597	1.6745	0.7000	0.3000	Overfitting
Calc Type	CNN Output Predictions - VGG16 - ReLU Sigmoid	0.0648	0.9963	0.0037	1.4531	0.6971	0.3029	Overfitting
Calc Type	K-fold Validation	0.0789	0.9956	0.0044	1.2262	0.7316	0.2684	Overfitting
Calc Type	Varying hyperparameters ELU Softmax	2.0249	0.5870	0.4130	2.0496	0.5176	0.4824	Underfitting
Calc Type	BatchNormalization Dropout	0.6415	0.9956	0.0044	1.5724	0.6941	0.3059	Overfitting

Fig 6: Summary of model performances

6. Evaluation

We evaluated other solutions found on Kaggle on this same version of dataset and compared the results with our VGG16 Setup.

Rank	Name	Test Loss	Test Accuracy	Source
1	Custom Setup	0.1352	0.9608	[Link]
2	**Our VGG16 Setup**	**1.2247**	**0.7324**	**[N]**
3	VGG16	2.3218	0.6853	[Link]
4	InceptionResNetv2	2.3876	0.6529	[Link]
5	EfficientNetB2	0.6777	0.5875	[Link]

Table 1: Comparison of Loss and Accuracy between different solutions

7. Conclusion

These models exhibit a need for a better balance between learning from the training data and generalizing to new data. The overfitting models could benefit from more regularization and data augmentation. For the underfitting model, a more complex architecture or feature engineering might be necessary to capture the dataset's nuances.

8. References

[N] Attached solution notebook CS6482-Assign1-23077751-23241977.ipynb

[1] CBIS DDSM Dataset

[2] Very Deep Convolutional Networks for Large-Scale Image Recognition

[3] Everything you need to know about VGG16