Topic: Develop First Modeling Approach Alice Tang and Yinda Chen

TF Lite Breast Cancer Detection Week 4: Data Pre-Processing

1. Model Approach

For this week, we chose to implement a Convolutional Neural Network (CNN) as our modeling approach for classifying breast cancer images. We were interested in using this model for several reasons related to the nature of the data, the prediction task, and the strengths of CNNs in image classification.

1.1 Rationale for the Choice

The dataset contains breast cancer images with complex patterns critical for accurate classification. CNNs are specifically designed to process visual data, and can automatically detect and learn features like edges, shapes, and textures from images. In turn, this reduces the need for extensive feature engineering.

CNNs have also demonstrated strong ability in medical imaging tasks, including breast cancer detection, through distinguishing subtle visual differences with high accuracy. This makes them a strong choice in detecting malignancies in breast cancer images. CNNs consistently outperform traditional image analysis techniques in both accuracy and reliability.

Additionally, CNNs scale well with larger datasets, allowing them to generalize better as they learn from more diverse data. Understandably, this scalability is an important aspect in medical applications, where datasets often vary in size and complexity. Considering our overarching goal to develop a TFLite application for detecting breast cancer, a CNN model could be a strong contender.

1.2 Complexity of the Modeling Approach

Implementing a CNN for breast cancer classification has several complexities. Firstly, training a CNN already requires significant computational resources, especially for large datasets or deep learning. Though GPUs can accelerate the process, resource limitations may be a challenge in certain environments. This is accurate for this week, as the code cannot be efficiently run in the JupyterHub environment. Instead, it was executed in the free Kaggle environment. The notebook can be found and run directly here. We are aware of this limitation, but since our goal is to develop a model for an application, we wanted to ensure we could train on the full dataset, even if it requires using a GPU.

Additionally, designing the CNN architecture includes selecting the number and type of layers (e.g., convolutional, pooling, dropout) and activation functions, all of which heavily impact model performance. Spending time in tuning hyperparameters such as learning rate and batch size will require careful and extensive experimentation for optimization.

Furthermore, CNNs are prone to overfitting, specifically with smaller datasets. This can be seen in our results of this week, which will be discussed below. Utilizing techniques like dropout, data augmentation, and early stopping may help with the issue of overfitting but not without adding complexity to the training process and requiring ongoing validation to maintain performance.

Lastly, evaluating the model performance involves using classification metrics such as accuracy, precision, recall, F1 score, and AUC-ROC. Naturally, these metrics are paramount in medical applications, where any errors could result in dire consequences. Cross-validation will be necessary to ensure the model's robustness— which further complicates the evaluation process.

Ultimately, we believe the CNN will be a good approach for classifying breast cancer images due to its ability to automatically learn and extract features from visual data. Even though there are multiple layers of complexities with computational requirements, architecture design, hyperparameter tuning, and evaluation, we believe that there is potential for high accuracy and insight into the data which makes CNNs a strong choice for this task.

2. Evaluating the Hyperparameters

Each hyperparameter we chose was selected for evaluation to optimize the CNN model's performance, improve generalization, as well as ensure effective training.

2.1 Learning Rate

The learning rate is crucial in determining the step size at each iteration while moving toward a minimum of the loss function. Essentially, if the learning rate is too high it may lead to divergence, whereas a rate that is too low will result in slow convergence. The goal is to find a striking balance between performance and efficiency. By using a learning rate of 1e-4 and implementing ReduceLROnPlateau callback— which reduces the learning rate when the metric stops improving— we can adjust the learning rate accordingly based on validation loss, which aids in stabilizing training and improving model performance.

2.2 Batch Size

We also chose to evaluate the batch size, which refers to the number of training samples that are utilized in one iteration. It is a valuable hyperparameter, as different batch sizes can directly affect the stability and speed of convergence. For example, smaller batch size can contain a more noisy gradient estimate, making optimization less predictable and slower to converge, but could improve generalization in the long run. In our models, we experimented with batch sizes of 16 and 32 to use across different models to evaluate their impact on training speed and model performance.

2.3 Dropout Rate

Dropout is part of a regularization technique where a certain percentage of neurons are randomly turned off in the neural network during training. Experimenting with different dropout rates (we used 0.4 and 0.5) supports the prevention of overfitting by influencing the network to learn more robust features.

This is especially important to implement in CNNs, where there is a high risk of overfitting, especially when the model complexity increases.

2.4 Number of Units in Dense Layers

The number of units (neurons) in the fully connected layers after the convolutional layers affects the model’s capability to learn complex patterns. We evaluated varying numbers of units (e.g., 128, 256, 512) which helps us assess how changes in model complexity affect performance on the validation dataset.

2.5 Regularization Techniques

L2 regularization is applied to the dense layers to penalize large weights and reduce overfitting by discouraging overly complex models. Evaluating L2 regularization (with a strength of 0.001) allows us to assess its effectiveness in generalization performance on unseen data.

2.6 Convolutional Layer Configurations

The number of filters and kernel sizes in the convolutional layers affect the model's ability to learn features from input data. Increasing the filters (e.g., 32, 64, 128, 256) enables learning more complex patterns, while varying the kernel size adjusts the network's receptive field.

2.7 Epochs

We also evaluated different numbers of epochs such as 10 and 20, to determine the optimal training duration where the model does not overfit. Monitoring the validation loss is essential for finding the balance between underfitting and overfitting.

Ultimately, evaluating these hyperparameters will allow us to achieve optimal performance on our model. Carefully considering the learning rates, batch sizes, dropout rates, units in dense layers, regularization techniques, convolutional layer configurations, and epochs are all critical in developing holistic models that perform well in generalization.

3. Model Performance Metrics

When evaluating model performance, we made sure to track several key metrics to determine how the model is learning and generalizing. By analyzing both accuracy and loss on the training and validation datasets, we can examine model performance and identify any potential issues such as overfitting. The following metrics were chosen for this evaluation:

3.1 Chosen Metrics

1. **Training Accuracy:** This metric indicates the percentage of correct predictions the model makes on the training dataset, throwing light on how effectively it is learning patterns from the data.

2. **Validation Accuracy:** This measures the percentage of correct predictions on the validation dataset, acting as a substitute for how the model is likely to perform on unseen data. It helps with assessing the model's ability to generalize.
3. **Training Loss:** This indicates the model's performance on the training dataset, where lower values indicate better performance. It demonstrates the difference between the model's predictions and the actual values.
4. **Validation Loss:** Similarly to training loss, this metric reveals how well the model is performing on the validation dataset. It is crucial for understanding if the model is overfitting or underfitting.

These metrics are important for evaluating model performance in classification tasks. Ideally, we're looking for high training accuracy combined with low validation loss— this would indicate a well-fitted model, while low validation accuracy and high validation loss may indicate overfitting.

3.2 Analysis of Training and Validation Metrics Across Variations

We adjusted the hyperparameters and created 3 variations of our CNN model.

Model 1

- **Training Accuracy** begins at **43.48%** and results in minimal improvement, reaching **45.54%** by epoch 10.
- **Validation Accuracy** is also relatively low and does not show significant improvement, only peaking at **45.57%**.
- **Training Loss** and **Validation Loss** begin high and improve very slowly, indicating the potential issues with learning capacity or data quality.

Model 2

- **Training Accuracy** shows there is a steady improvement from **49.26%** to **90.32%** by epoch 20.
- **Validation Accuracy** also improves significantly, peaking at **58.33%** in the later epochs. However, it is still noticeably lower than training accuracy, indicating that there is potential overfitting.
- **Training Loss** decreases steadily from **8.82** to **0.87**, while **Validation Loss** also decreases but remains higher than training loss, suggesting overfitting may be occurring.

Model 3

- **Training Accuracy** shows a strong upward trend from **46.75%** to **90.36%**, indicating improved learning.
- **Validation Accuracy** also rises, the highest being **58.94%** by epoch 20, which indicates there is slightly better generalization than Model 2.

- **Training Loss** decreases significantly from **9.08** to **0.59**, and **Validation Loss** follows a similar trend. Because it is still higher than training loss, there may be potential overfitting here but less severe than the previous models.

3.3 Comparison Table of Each Model

Model	Epoch	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
Model 1	1	0.436575532	0.430272102	5.737282276	2.272104979
Model 1	2	0.463469297	0.443877548	2.054833889	2.275307894
Model 1	3	0.448229492	0.454081625	1.865471601	1.768943787
Model 1	4	0.447781265	0.43877551	1.784791708	1.721961498
Model 1	5	0.462572843	0.431972802	1.735982418	1.762647152
Model 1	6	0.472882122	0.426870733	1.746888399	1.669961214
Model 1	7	0.459883451	0.459183663	1.68084991	1.642323852
Model 1	8	0.453608245	0.421768695	1.648337841	1.803120136
Model 1	9	0.440161377	0.416666657	1.655378222	1.616591215
Model 1	10	0.458090544	0.431972802	1.599865913	1.587667823
Model 2	1	0.46884805	0.13945578	9.956824303	16.33875275
Model 2	2	0.572837293	0.13945578	2.043967247	28.35334015
Model 2	3	0.642761111	0.13945578	1.868190527	31.95510864
Model 2	4	0.678619444	0.13945578	1.725112557	32.53997421
Model 2	5	0.69520396	0.13945578	1.687874556	27.54066658
Model 2	6	0.716718972	0.222789109	1.589014292	11.76173496

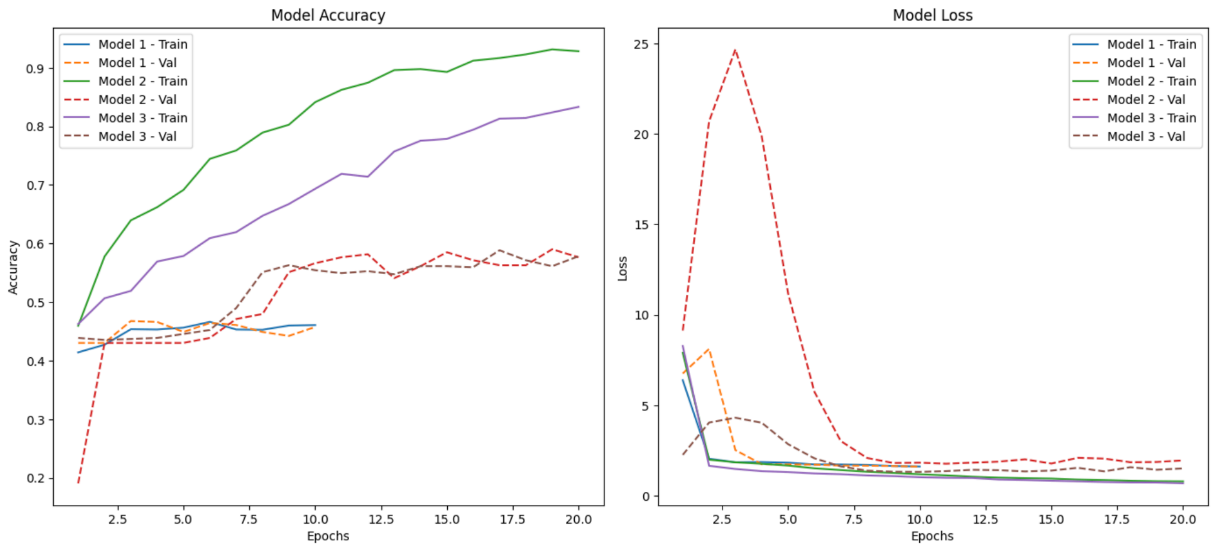
Model 2	7	0.748543262	0.401360542	1.498165965	4.066748142
Model 2	8	0.778126419	0.552721083	1.411539078	2.049757004
Model 2	9	0.807261288	0.556122422	1.324633598	1.828861833
Model 2	10	0.821156442	0.574829936	1.283990264	1.875239134
Model 2	11	0.857911229	0.540816307	1.189439416	1.81374836
Model 2	12	0.872702837	0.578231275	1.149752021	1.880964637
Model 2	13	0.869116962	0.542517006	1.108883739	1.824545741
Model 2	14	0.868668735	0.571428597	1.088928819	2.052918434
Model 2	15	0.885253251	0.568027198	1.040731907	1.954127312
Model 2	16	0.891976714	0.568027198	1.028036594	2.089831829
Model 2	17	0.889287293	0.561224461	0.99992317	2.060806513
Model 2	18	0.906768262	0.571428597	0.953275919	2.145747662
Model 2	19	0.904975355	0.576530635	0.913313866	1.999648571
Model 2	20	0.912147045	0.556122422	0.897244751	1.875790119
Model 3	1	0.475571483	0.430272102	7.476365566	2.061024904
Model 3	2	0.50649935	0.430272102	1.710907578	2.782779455
Model 3	3	0.555804551	0.426870733	1.443171263	2.580574512
Model 3	4	0.588077068	0.447278917	1.333048224	2.159253836
Model 3	5	0.612729728	0.455782324	1.261053324	1.810233235
Model 3	6	0.663827896	0.481292516	1.174334764	1.59027493
Model 3	7	0.672792494	0.481292516	1.113182187	1.496538162

Model 3	8	0.71268487	0.508503377	1.061157227	1.574999809
Model 3	9	0.718960106	0.559523821	1.011063337	1.328755379
Model 3	10	0.763334811	0.545918345	0.939538836	1.344109058
Model 3	11	0.784849823	0.576530635	0.880687833	1.446696162
Model 3	12	0.797400296	0.549319744	0.832355797	1.432986498
Model 3	13	0.796503782	0.505102038	0.830967247	1.566656351
Model 3	14	0.816674113	0.542517006	0.779839039	1.565706611
Model 3	15	0.831017494	0.544217706	0.752505779	1.573444963
Model 3	16	0.848946631	0.557823122	0.71983093	1.57783711
Model 3	17	0.844912589	0.569727898	0.707660258	1.544944167
Model 3	18	0.845809042	0.549319744	0.672147155	1.567885637
Model 3	19	0.870013475	0.56292516	0.636104047	1.703684449
Model 3	20	0.873599291	0.583333313	0.61708039	1.62181735

3.4 Comparison of Validation Dataset Performance Metrics

- From this, we can confirm that Model 1 has the lowest validation accuracy (max 45.57%) and highest validation loss (max 5.44). This model notably struggles to generalize.
- Model 2 shows decent improvement in comparison to Model 1 in validation accuracy (58.33%) and a validation loss of 1.73, thereby exhibiting a more effective learning and generalization process but still struggling with overfitting.
- Model 3 has the highest validation accuracy (58.94%) and a validation loss of 1.68, it outperforms both Model 1 and Model 2 in terms of validation metrics. This shows us that changes made in this model's configuration effectively enhanced performance.

3.5 Accuracy and Loss Function Across All Models



The progression of training and validation metrics across the three models emphasizes the need to evaluate both for a better understanding of a model's learning behavior and generalization ability. While Model 2 shows substantial improvements in validation metrics, the most balanced performance is observed in Model 3— suggesting successful tuning and potential readiness for deployment or further testing. However, we would like to note that this CNN is simply a baseline model. It can be further improved by experimenting with deeper architectures, continuing to fine-tune hyperparameters such as learning rate and batch size, and experimenting with more optimization techniques Additionally, if other future models do not perform as well, we would be interested in exploring pre-trained models like residual networks (ResNets) that could enhance the model's performance and generalization. However, we decided to build a CNN model on our own for this week because it gave us more control over the architecture and interpretability over the layers.

4. Model of the Week: Best Performance

Based on the analysis of the training and validation metrics across the three models, we can affirm that **Model 3** is the best model for this week. Below is a summary of the reasons for this choice:

- **Training Accuracy**: Model 3 exhibits a significant increase in training accuracy, starting at **46.75%** and reaching **90.36%** by epoch 20. This increase shows that the model is effectively learning from the training data.
- **Validation Accuracy**: The validation accuracy of Model 3 is highest at **58.94%**, this is the highest among all three models. This reveals Model 3 is better at generalizing to unseen data

compared to Model 1 and Model 2, which had lower validation accuracies (45.57% and 58.33% respectively).

- **Training Loss**: Model 3 has a low training loss of **0.59** by epoch 20, showing that the model's predictions align with the actual truth in the training data.
- **Validation Loss**: The validation loss for Model 3 is **1.68**, which is lower than both Model 1 and Model 2. Obtaining a lower validation loss shows that the model is performing better on the validation dataset, which is essential in assessing its generalization capability.

Overall Model 3 shows that there is a better balance between training accuracy and validation accuracy. Even though Model 2 had a higher training accuracy, the validation accuracy did not improve much, indicating overfitting. If we look at each model holistically, Model 3's validation metrics reflect a more robust model that is less likely to overfit. Thus, Model 3 is selected as the best model for the week due to its high training and validation accuracies, low training and validation losses, and better generalization to unseen data.

Though there are still significant improvements to be made for optimizing performance and ensuring reliability, we consider this a solid benchmark to build upon. Additionally, while the code cannot currently run on JupyterHub due to memory constraints when uploading the entire dataset, it works smoothly and for free in Kaggle's environment. Moving forward, we will continue thinking of ways to adapt the code to JupyterHub despite the environmental limitations. However, since our ultimate goal is to develop a reliable TFLite app for detecting malignancies in mammography images, we prioritized training the model on the full dataset, even if it meant temporarily being unable to run on JupyterHub.