Programming Assignment 3: Evaluating Deep Learning Models

Adam Torek
Professor: Jun Zhuang
CS 597: Special Topics in Deep Learning
Spring 2025 Semester

Part 1: Explaining Evaluation Metrics For Deep Learning Models

Accuracy: This metric measures how many labels are correctly predicted by a model vs the total number of labels in the dataset. This metric can be used with multi-label as well as binary data. However, this metric can be skewed because of imbalanced labels, resulting in results that can look better or worse than they actually are. Accuracy itself does not measure the true positive, true negative, false positive, or false negative rate of a given binary classification dataset. Accuracy is consistently used alongside other metrics such as the ones below.

Precision: Measures the number of true positives labeled by a model divided by the sum of the false positives plus true positives. This metric is used for binary classification tasks and shows how good a given model is at avoiding false positives. A precision score of 1.0 would indicate all of the positives labeled by the model are true positives, thus showing it is correctly able to label all of the positive examples in a given dataset. Conversely, a precision score of 0.0 would show the model is completely unable to predict true positives in a given dataset over false positives.

Recall: A measure of the sum of true positives divided by the sum of true negatives and false positives. Like precision, this metric is commonly used in binary label classification tasks and measures how many correct positives are predicted over the total number of positives possible. A model with a recall score of 1.0 means that it is correctly able to identify 100% of all true positive examples without any false negatives. A recall score of 0.0 indicates the opposite, showing a given model cannot correctly predict any true positives from the set of all possible positives.

F1-Score: This is a harmonic mean of both precision and recall. This metric also commonly used in binary classification tasks alongside precision and recall separately. The F1-score gives a balanced mean between the individual precision and recall scores, giving this metric more stability than precision and recall alone. In highly imbalanced binary classification tasks, the F1-score can serve as a more reliable substitute for accuracy.

ROC Curve and AUC: The ROC, or receiver operating characteristic curve, is used to measure the true positive rate vs the false positive rate of a given model at different selected intervals. The true positive rate and false positive rates are calculated for each interval and graphed against each other in a curve ranging from 0.0 to 1.0 for both the true positive rate and false positive rate. The ROC curve measures the effectiveness of a given model at different positive and negative rates, allowing an overall comparison across all chosen thresholds. The AUC stands for area under the curve, which is the total area under the ROC curve.

The AUC measurement turns the ROC graph from an array of metrics into a single number, showing how good a given model is at correctly finding true positives and true negatives all chosen intervals. An AUC of 1.0 is the best possible value, indicating that model can correctly label all false positive and negative samples across all possible selected ratios of a given set of negatives and

positives. Conversely, an AUC of 0.0 is the worst case scenario, showing that the model cannot label any negatives and positives correctly at any selected split of positives to negatives across the dataset.

Part 2: Evaluating Deep Learning Models on an Image Classification Task

Sub-Part 1: Experimental Setup

I chose image classification as the task I will train and evaluate my deep learning models on. I selected this task because it is a well established machine learning task with a wide variety of datasets and models available for use. In fact, PyTorch provides a range of image classification models and datasets ready to go within its TorchVision package, which is in part why I chose this task. I also chose image classification because it is a very practical task with a wide variety of uses including data labeling, optical character recognition, detecting objects for self-driving cars, and so on. To make my evaluation simpler and use more evaluation metrics to compare my models, I decided to focus on binary image classification.

I am also choosing three different models for evaluation on my binary image classification task. These models are the following:

- MobileNet-V3-Large: This is a relatively compact and low-parameter convolutional neural network (CNN) meant to be used on resource-constrained devices such as mobile phones (hence the name MobileNet). Despite being small, MobileNet-V3 and its predecessors have proven effective for image classification and other computer vision tasks. It is comprised of a series of bottleneck layers which split an image into many smaller feature maps which are then upscaled and sent through many small convolution filters before being combined at the end of the bottleneck layer. I chose to evaluate the large version of this model because it has proven to be the most effective and parameter-efficient of the MobileNet family overall, and because MobileNet goes the opposite way of most other models, sacrificing performance for smaller parameter counts.

- Swin-Tiny: This model is a vision transformer, unlike the other two models, which are CNNs. Swin takes the transformer architecture, commonly used for text and sequence-based tasks, and applies it to computer vision and image classification instead. Swin uses a series of transformer blocks with multi-head attention layers at the base and multi-layer perceptrons above them. What makes Swin Transformer unique is that it splits a given image into variously sized patches between layers that grow in size from the first layers to the last layers, allowing different transformer blocks to slowly go from local image patterns to global ones. I chose Swin-Transformer because it is a transformer model set up to slowly extract features from images like a CNN, theoretically giving the benefits of both models. I specifically chose Swin-Tiny because it is the fastest and easiest to train of the Swin family. Despite that, it is much more parameter-heavy than the other two models due to it being a transformer.

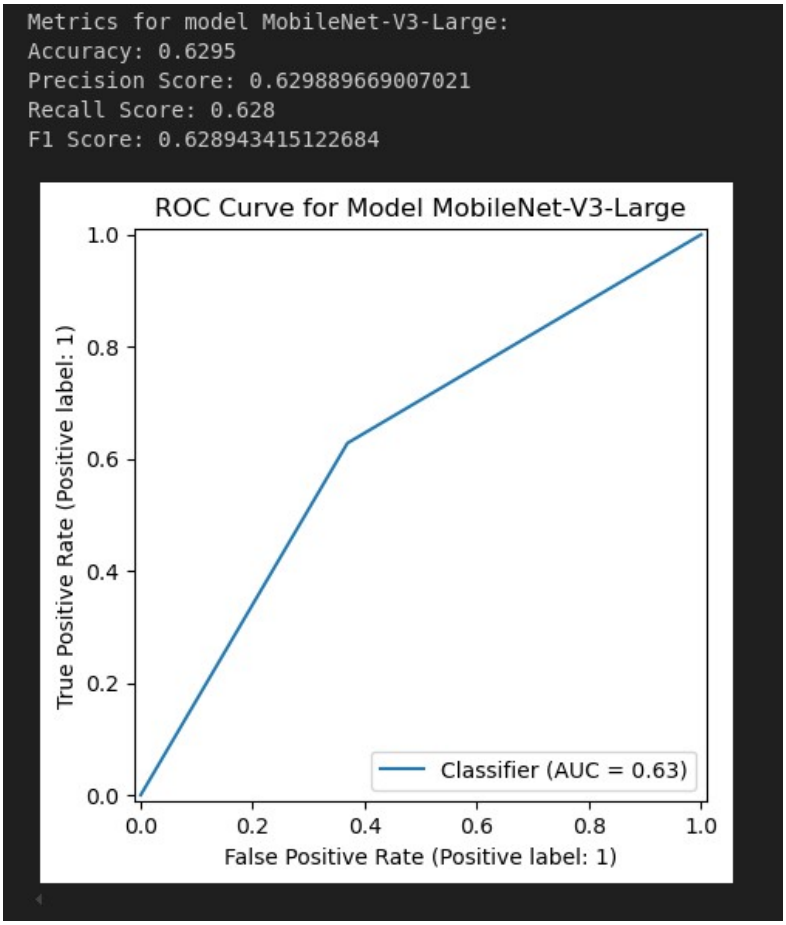
- ResNet-50: This medium-sized convolutional neural network (CNN) was first established in 2015 for the ImageNet image classification contest. ResNet is unique because it makes extensive use of residual connections between convolutional layers, where input from one feature map is carried forward to the output of another feature map without being modified. These residual connections allow ResNet to have a kind of “memory” between its feature maps, thus allowing it to be more parameter efficient. I chose ResNet-50 because it is the most

efficient model from the ResNet family overall, and because I wanted to see if extensive residual connections helped on the binary image classification task compared to other models.

For this task, I chose the CIFAR10 dataset, a commonly used benchmark for measuring model effectiveness in image classification. I specifically chose the first two labels of the dataset to convert it to a binary classification task. I then added a binary classification head on top of the trained parameters of MobileNet-V3-Large, ResNet-50, and Swin-Tiny. I froze the base weights of these models to make the training faster and because the impact on evaluation performance was negligible. The binary classification heads of these models was still trained. To make my results comparable, I used the AdamW optimizer set to a learning rate of 0.0001, a batch size of 32, and 10 training epochs for all the models. I used the binary CIFAR training set to train the models and the same binary labels in the CIFAR test set for evaluation. I used accuracy, precision, recall, f1-score, ROC curve, and AUC score for all models for a thorough and complete comparison across all models.

Sub-Part 2: Evaluation Results

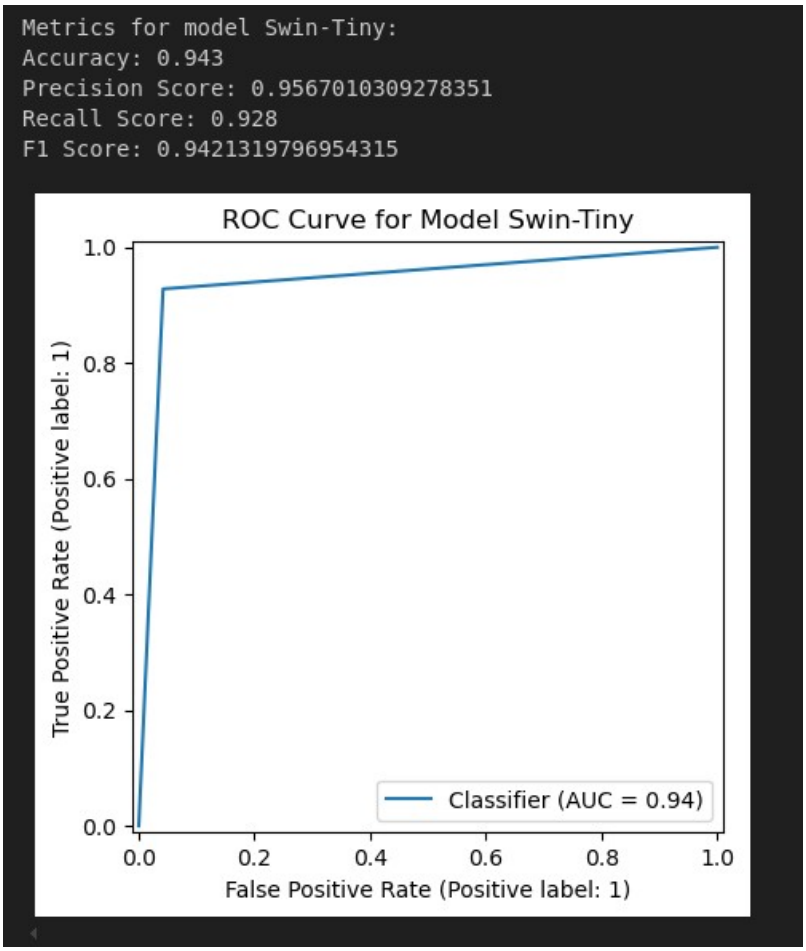
Results for MobileNet-V3-Large:



Overall, MobileNet-V3 did the worst out of all the models I tested. Even its large version gets bad results across the board, including accuracy, precision, recall, F1-score, the ROC graph, and the AUC score. All of these values are between 0.628 to 0.63, may points behind both Swin-Tiny and ResNet-50. I think this is because this model is overall small and relies heavily on bottleneck layers,

which while parameter-efficient, are not as capable of exploiting local-to-global patterns as either Swin-Tiny or ResNet-50. MobileNet-V3 is also a small model overall, which is likely causing an underfitting problem where it cannot properly learn or express deep features hidden within its training data.

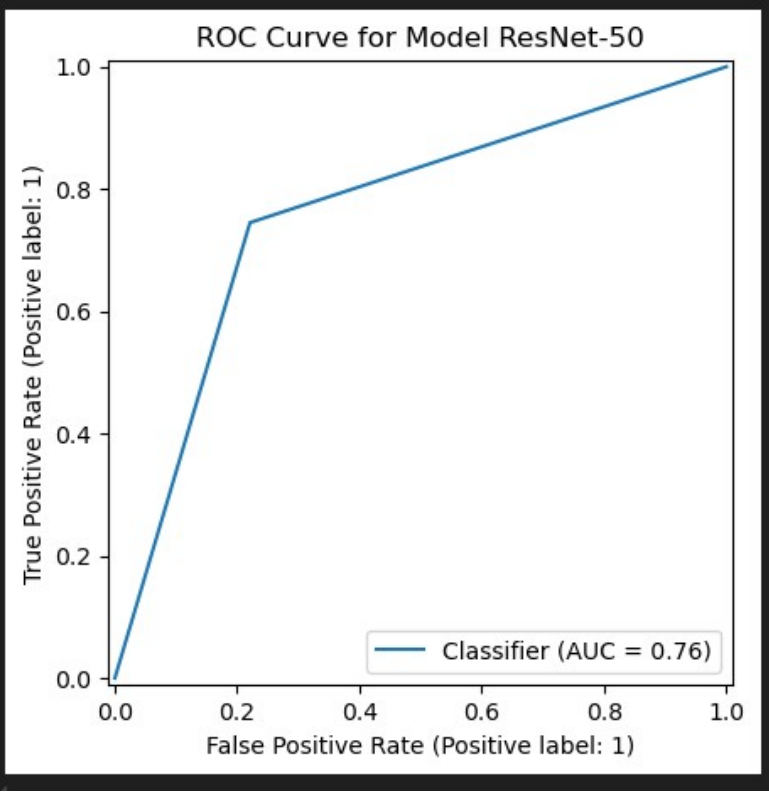
Results for Swin-Tiny:



Swin-Tiny did the best out of the three models by far. It's accuracy, precision, recall, F1-score, and AUC are all in the range of 0.928 to 0.958, indicating this model is very good at correctly labeling true positives and true negatives. I believe Swin-Tiny did the best because it is a transformer model that gradually splits a given image into larger and larger patches. The attention heads in Swin-Tiny likely allow the model to learn long distance context between pixel values, and the gradually increasing size of the image patches across layers allow the model to exploit local-to-global patterns similar to the repeating feature maps of a CNN. Both of these factors combined are likely the reason why it performs about 0.3 points ahead of MobileNet-V3 and about 0.15 points ahead of ResNet-50 across all evaluated metrics.

Results for ResNet-50:

```
Metrics for model ResNet-50:
Accuracy: 0.762
Precision Score: 0.7712215320910973
Recall Score: 0.745
F1 Score: 0.757884028484232
```



Overall, ResNet-50 did better than MobileNet-V3-Large across all evaluated metrics, but worse than Swin-Tiny. ResNet-50's size and heavy use of residual connections allow the model to retain relevant feature map information across layers and exploit local-to-global pattern matching, which likely contributes to its increased performance over MobileNet-V3-Large and its bottleneck layers. ResNet-50 is also larger than MobileNet-V3-Large, giving it more parameters that can learn more features overall. Despite outperforming MobileNet-V3-Large across all evaluation metrics, ResNet-50 is still heavily outperformed by even smallest Swin Transformer model, Swin-Tiny. This is likely because it's residual connections, convolutional feature maps, and pooling layers cannot hold as much modeled information as Swin's transformer layers (especially its attention heads) or exploit global-to-local connections quite as effectively as Swin-Tiny can.

Sub-Part 3: Potential Shortcomings Of Metrics

Overall, all of my metrics have certain weaknesses that in some cases can be overcome but in some cases cannot be. Accuracy is vulnerable to imbalanced datasets where one label or set of labels has a much higher presence than other labels in the dataset. For example, an highly balanced dataset with 900 positive samples and 100 negative samples may cause a model's accuracy to lie at 0.905, which may seem high, but is in fact barely above the dataset's skew of positive to negative samples. A

way to mitigate this would be to establish a baseline accuracy and demand that any trained deep learning model must get an accuracy score above that baseline to even be considered worth using. Another way to combat accuracy issues is to make the dataset balanced, either through oversampling, undersampling, or generating synthetic data. Another option is to just ignore accuracy and use another metric such as the F1-score instead.

Precision and recall may also be vulnerable to this imbalance issue. If there are too many positives and too few negatives or vise versa, the model may appear to be performing better or worse than it really is. The F1-score can act as a stabilizer from either precision or recall leaning too far in one direction by acting as a mean between the two. However, if the dataset is extremely imbalanced, even the F1-score may be vulnerable to this skew. In such cases, oversampling, undersampling, or synthetic data generation may be used as a last resort.

Lastly, a problem with all of these metrics themselves is that they may not fully convey the impact of a false positive or false negative in a real-world use case. Take cancer screening as an example. A false positive would diagnose someone with a particular form of cancer that does not have one, however a false negative is not diagnosing someone with that form of cancer when they in fact have it. The impact of a false negative is far worse than the impact of a false positive in this case. A false positive patient will be seen for follow up examinations and diagnostic testing, while a patient with a false negative will unknowingly avoid treatment because both they and their care provider may continue to be unaware of the cancer. This could lead to severe complications or even death. In order to avoid this issue, any users, developers, researchers, and contributors to deep learning technology must be aware of the risks of their particular task and adapt accordingly. Only they can give meaning to the impact of these scores in their particular use case and use caution before using deep learning models in production or real-world environments.