ML Take-Home Assignment, Autify, Inc. technical assignment for the Senior Machine Learning Engineer, LLMs & Prompt Engineering (New Project) position!

Hari Vidharth

April 2024

Problem Statement and Initial Observations

In modern web applications, the standard checkbox elements are often replaced with complex and customized designs, making traditional detection methods insufficient. This document outlines my thought process for solving this challenge using a deep learning-based system aimed at detecting and classifying checkboxes as checked, unchecked, or other.

My initial observations are there are a total of 515 images. In the context of deep learning the amount of image data is limited. Also, another observation is there is a data imbalance among the three classes.

Model Selection and Baseline Setup

To establish a baseline, three state-of-the-art deep learning models (ResNet, DenseNet, EfficientNet) are chosen for image classification. All models are initialized with equal or default parameters and trained without pretraining. Training metrics such as loss, accuracy, and training time are recorded. For example, no augmentations, image size 244 x 244, batch size 32, same loss function which is cross-entropy loss, 200 epochs, the only difference being the optimizers where RESNET & DENSENET use SGD while EFFICIENTNET uses RMSProp this is to be consistent with the original research paper. To ensure consistency and reproducability the torch global seed is set to 101. Even though deeper versions of the models are available the base models have been chosen for this use case.

- RESNET: Epoch 178/200, Train Loss: 1.7634, Train Accuracy: 0.9757, Val Loss: 0.6539, Val Accuracy: 0.7843

- DENSENET: Epoch 194/200, Train Loss: 6.0504, Train Accuracy: 0.8252, Val Loss: 0.7735, Val Accuracy: 0.7059
- EFFICIENTNET: Epoch 112/200, Train Loss: 0.5425, Train Accuracy: 0.9854, Val Loss: 0.6787, Val Accuracy: 0.7843

Without pretraining, all models exhibit comparable performance, with EfficientNet showing faster convergence. However, training from scratch on limited data raises concerns about model complexity and overfitting.

PreTraining and FineTuning

Pre-trained weights from ImageNet are used, and models are fine-tuned on the custom data. Here there are two ways to approach this problem the first is to freeze the layers and only train the final layer on the new data. This approach works fine if the previous and new data are similar to each other. But I believe there are lots of differences between IMAGENET data and this one as this is a very specific use case of a given scenario so the best choice was to fine-tune the model with all the layers active and lower the learning rate by a little with the other being either the same from the previous step or remaining as default.

- RESNET: Epoch 186/200, Train Loss: 0.2301, Train Accuracy: 1.0000, Val Loss: 0.5844, Val Accuracy: 0.7843, Test Accuracy: 0.7115
- DENSENET: Epoch 118/200, Train Loss: 0.3567, Train Accuracy: 1.0000, Val Loss: 0.5840, Val Accuracy: 0.7647, Test Accuracy: 0.7308
- EFFICIENTNET: Epoch 96/200, Train Loss: 2.9490, Train Accuracy: 0.9393, Val Loss: 0.6296, Val Accuracy: 0.7843, Test Accuracy: 0.6538

Based on baseline and fine-tuned results, EfficientNet is selected as the primary model for its scalability, efficiency, and faster convergence rate. It balances model depth with fewer parameters, making it suitable for limited data scenarios. One more observation here is now the other models seem to be overfitting with high training accuracy but lower validation accuracy while the EFFICIENTET model seems to be slightly underfitting maintaining a decent training-to-validation accuracy ratio which can be improved upon by further tweaking and tuning.

Next Steps & Improvements

- Further decrease the learning rate to 0.001 (From 0.265 to 0.01 to 0.001) to facilitate better convergence.
- Incorporate LookAhead optimizer with RMSProp for improved optimization.

- Enhance image resolution from 224 to 299 to capture finer details.
- Introduce auto augmentations based on ImageNet training data and normalize images using mean and standard deviation from ImageNet.
- Experiment with weighted cross-entropy to address class imbalance.
- Implement custom augmentations tailored for checkbox detection.
- Custom dataset based image normalizations.

Performance Evaluation

- Checkpoint 1: Used IMAGENET Auto augmentations which yield significant performance gains in terms of validation and test accuracy. Epoch 32/100, Train Loss: 1.1021, Train Accuracy: 0.9733, Val Loss: 0.3288, Val Accuracy: 0.9216, Test Accuracy: 0.8846.
- Checkpoint 2: Normalizing images with augmentations also contributes to improved accuracy but less significantly compared to augmentations. Epoch 74/100, Train Loss: 1.1939, Train Accuracy: 0.9806, Val Loss: 0.3920, Val Accuracy: 0.9020, Test Accuracy: 0.7692
- Checkpoint 3: Combining augmentations and normalizations leads to the highest performance observed so far. Epoch 29/100, Train Loss: 1.0295, Train Accuracy: 0.9684, Val Loss: 0.2151, Val Accuracy: 0.9216, Test Accuracy: 0.8846
- Checkpoint 4: Weighted cross-entropy slightly reduces training loss but may lead to underfitting. Epoch 65/100, Train Loss: 0.3489, Train Accuracy: 0.9927, Val Loss: 0.3663, Val Accuracy: 0.9412, Test Accuracy: 0.8462
- Checkpoint 5: Custom augmentations with hyperparameter tuning and without class balancing exhibit good performance but not as high as the previous approach. Epoch 88/100, Train Loss: 0.3277, Train Accuracy: 0.9951, Val Loss: 0.3522, Val Accuracy: 0.9020, Test Accuracy: 0.8654

Conclusion & Future Work

EfficientNet serves as a robust foundation for checkbox detection, offering scalability and efficiency. By incorporating various enhancements, including augmentations, normalizations, and class imbalance handling, significant improvements in accuracy and convergence rate are achieved. The combined approach of auto augmentations and normalizations proves to be the most effective, striking a balance between performance and computational efficiency which has been used as the final model to be submitted for inference. Here the main idea was to balance training time, convergence rate, overfitting on limited data, model complexity, and system and memory resources while still trying to achieve and squeeze out the maximum performance. The idea behind model selection had a deployment/user perspective as well when it comes to automated software testing software remotely. This could be discussed more in detail later on.

Some ideas for future work include The obvious one would be to bring in more training data. Improve model complexity by using higher-end models than the base models used here. Do image-specific study and analysis and image-specific augmentations instead of overall dataset based augmentations.