

Fine Grained Image Classification

Team: Zesheng Liu, Xinxin Liu, Ruilai Xu. **Project Mentor TA:** Anish Bhattacharya

GitHub: <https://github.com/StarLiu714/Cassava-Fine-Grained-Image-Classification>

1) Abstract

In this project, we aim to classify Cassava Leaf Diseases using a dataset of labeled images provided by the Cassava Leaf Disease Classification Challenge. We implemented a wide range of different approaches to classify the diseases using deep learning models, including traditional models like VGG19 and ResNet, and advanced models like EfficientNet and Vision Transformers. We fine-tuned the models using various hyperparameters, loss functions, optimizers, and data augmentation techniques, and evaluated them based on accuracy on a held-out validation set and on the Kaggle Leaderboard. Our results show that advanced models like EfficientNet and Vision Transformers can achieve high accuracy with less computational resources, and that data augmentation techniques and optimizing hyperparameters, loss functions, and optimizers can significantly improve the accuracy of all models. Our experiments demonstrate the importance of careful selection of data augmentation techniques and hyperparameters for effective model performance on this task, and that ensembles can further improve performance scores. Overall, our project provides insights into the effective implementation of deep learning models for the classification of Cassava Leaf Diseases.

2) Introduction

Set up the problem: Cassava, a major staple food crop in Africa, is affected by various leaf diseases that can lead to significant yield losses. In this study, we aim to classify Cassava Leaf Diseases using a dataset provided by the Cassava Leaf Disease Classification Challenge. The dataset consists of 21,397 images, with five possible classes. Our machine learning model will take an input image and output the corresponding disease class.

In our project, we will focus on classifying the disease label for those leaf images. Our input will be a combination of image data. Our raw images are in RGB format and the size is 600×800. Our model output will be probability values representing the probability of image belonging to different classes.

Implementation: We will implement different kinds of models, which includes the baseline CNN model, traditional model: VGG and ResNext, and advanced model: Bilinear CNN, ResNext, EfficientNet and Vision transformer. In implementation, we will fine-tuning the model for the best accuracy, considering different loss functions, optimizers and different LR scheduler. Depending on the model architectures, we also explore different data augmentation methods and choose the best augmentation pipeline for different models.

Evaluation: In our evaluation, the main factor is the accuracy of prediction, which is evaluated either on a held-out set from the whole given dataset or through the Kaggle Leaderboard. In the training process, we tried different image augmentation methods and evaluated their influence on final accuracy. We also tried to build different combinations of individual models as ensembles, and evaluate their performance.

2) How We Have Addressed Feedback From the Proposal Evaluations

-Decision about whether to train from scratch or use pretrained weights: We finally always use the pretrained weights except for the baseline, as we find that if we train it from the very beginning, it will take a longer time and we can't get high accuracy with that.

-Fine-tuning the training process: We tried different learning rates for different models, and tried several different LR schedulers to improve the training process and avoid overfitting. We also use the confusion matrix as a helpful tool to decide the choice of augmentation methods.

3) Background

For fine-grained image classification, there are two kinds of methods: One is using the general image classification method(not limited to fine-grained classification) and fine-tuning the hyperparameter to get a reasonable accuracy. The other is taking the pixel spatial information into consideration and probably even consider some marked feature points and bounding boxes, then design new architectures for both localization and classification. There is no common rule that which method will have the best performance for a specific dataset.

Our project aims to address the shortcomings of these prior works by comparing and fine-tuning one fine-grained classification model and several general image classification models of latest fashion, improve performance under limited computational resources, find out the better models, and then put the relatively better models into an ensemble. Then, by applying such an ensemble, we expect to enhance the model's generalizability and robustness. Additionally, by data augmentation methods, including different cropping techniques and varying contrast, we expect to improve model's performance in real-world applications handling unseen dataset.

The relative works (model, paper, and kaggle submissions) are as follows: VGG19[1], ResNet[2], ViT[3,6], EfficientNet[4,7], ResNext[5,8]. Among the above, the most relevant work to our project is:

A. <https://www.kaggle.com/competitions/cassava-leaf-disease-classification/discussion/207450>

It is a few steps for starting, to give out some idea about what we can explore for this dataset and aspects that we can consider to improve the model performance.

4) Summary of Our Contributions

Implementation contribution(s): We built a pipeline for data loading and data augmentation, then we tried different models, fine-tuning and testing for their performance. All the models we have tried are: baseline CNN, VGG19, ResNet34d, ResNet50d, ResNext50_32x4d, ViT_base_patch16_384, Bilinear-CNN, EfficientNet-B3, and EfficientNet-B4. We also tried to combine our best individual models to ensembles and see how it works.

Evaluation contributions: We evaluate our model by two methods. One is through a held-out testset of our whole image dataset. We held 30% of them to be a test set and get the corresponding test accuracy. In order to better see how models perform for different classes, we drew the confusion matrix to help understand and fine-tuning the augmentation part. The other is through the Kaggle LeaderBoard. We submit our best individual models to the leaderboard and get the score of performance, which represents the accuracy on a private testset. In the last, we combine different models to form an ensemble and compare the results with single models.

5) Detailed Description of Contributions

5.1 Implementation Contributions

5.1.1 Feature Engineering

First we did basic feature engineering and exploration of the dataset. There are mainly two things that need to be noticed:

- *Data imbalance*: It is clear that class 3, "Cassava Mosaic Disease (CMD)", is the majority, which accounts for more than half of the whole dataset. An imbalanced dataset may cause the model to learn mostly from the majority class. After doing research with Kaggle discussion and several best solutions, we finally form the idea that it represents the distribution of real world situations and prior experiments shows that formally upsample the number of other classes wouldn't help a lot in final accuracy. Most models show great robustness to the imbalance data. By changing the loss function to Taylor Cross Entropy, Bi tempered Logistic Loss and adding label smoothing, we can also handle this problem and make the model converge faster.

- *Image size and Noise*: The original image size is 600×800 pixels, which is quite big. In order for a higher training speed, we will reduce the image size by either directly resizing the original image or do random crop. Besides, we notice that raw images have some background noise, so reducing the image size can also be helpful to avoid the influence of noise.

5.1.2 Model Implementation

In our project, we implemented a wide range of different approaches to classify cassava leaf diseases using deep learning models. We systematically compared the performance of these models, varying hyperparameters such as loss function, optimizer, learning rate scheduler, data augmentation techniques, batch size, and image size.

Our key implementation contributions included:

Implementing a baseline multi-layer CNN model, which served as a starting point for our comparisons with more advanced architectures. This is a simple baseline model that uses a multi-layer convolutional neural network (CNN) architecture to classify the Cassava Leaf Diseases. It doesn't have any special features or architectures but is a good starting point for classification task.

Implementing VGG19[1] and ResNet[2] architectures, two popular CNN-based architectures frequently used in image classification tasks. VGG19 has a large number of trainable parameters and is capable of capturing fine-grained details in images. ResNet50 architecture, which is known for its residual connections and ability to train very deep neural networks in a shorter time without huge computational resources. ResNet50 is capable of capturing both low-level and high-level features in images, making it a good choice for image classification tasks. These two models are also frequently used as backbone of some complex models, aiming for feature extraction. Therefore, it is necessary to implement and see their performance.

Implementing advanced models in image classification, such as Bilinear CNN[10], ResNext[5,8], EfficientNet[4,7], and Vision Transformer (ViT)[3,6] models. These models are known for their superior performance on large-scale image classification tasks and have been successful. EfficientNet architecture is known for its efficiency and good performance in image classification tasks. EfficientNetB3-NS uses the pretrained weights by NoisyStudent, which is a semi-supervised learning method and is shown to be more robust and can improve its accuracy on all EfficientNet families.

Bilinear CNN uses ResNet or VGG as backbone and introduces the Bilinear pooling layer, which is known for its ability to capture complex interactions between features in images and relationships between different parts of the image. ResNeXt50 has multiple parallel paths through the network, which allows it to capture more diverse features in images. And Resnext50_32x4d+Taylor is based on the ResNeXt50 architecture like the previous model, but uses a different loss function, Taylor Cross Entropy with Label Smoothing, which has been shown to be more robust to noisy labels. Vision Transformer (ViT) architecture, which is a relatively new type of neural network that uses self-attention mechanisms to capture relationships between different parts of the image. This model uses the Bi-Tempered Logistic Loss function and is trained on larger images, which allows it to capture more fine-grained details and be more robust for noise in image and labels. And we used a different learning rate scheduler, Cosine Annealing Warm Restarts, which has been shown to improve performance on image classification tasks.

Varying loss functions include Cross Entropy, Taylor Cross Entropy with Label Smoothing, and Bi-Tempered Logistic Loss. Varying optimizers to include Adam and AdamW with different learning rates and weight decay. Implementing learning rate schedulers such as Reduce LR On Plateau, CosineAnnealingWarmRestarts, and WarmRestarts. For image augmentation, besides the normalization process for training and test set, we applied various data augmentation techniques for cropping, resize, flip, rotation, contrast, dropout and saturation values., building a preprocess pipeline for images. Varying the batch size and image size of the input data can either make us be able to train larger models, and test the effects of these parameters on the model's performance.

We systematically compared the performance of these models on the Cassava Leaf Disease Classification Challenge dataset, evaluating them based on accuracy on a held-out validation set, which is 30% percent of the whole given dataset. Attached is the detailed sheet of our models including: Model, Loss Function, Optimizer, scheduler, Train Augmentation, Batchsize, Image size and Accuracy are shown in the table.

[illegible]

5.2 Evaluation Contribution

The key question our experiments aimed to answer was which deep learning models and hyperparameters are most effective for classifying cassava leaf diseases, and how much improvement can be achieved over a baseline multi-layer CNN model. We also wanted to explore the factors that contribute to the superior performance of certain models over others, such as the effects of different data augmentation techniques and the tradeoffs between batch size and image size. How do different machine learning models compare in terms of accuracy in classifying Cassava Leaf Diseases? How do different hyperparameters, loss functions, optimizers, and data augmentation techniques affect the accuracy of machine learning models in classifying Cassava Leaf Diseases?

Our performance metric is accuracy, which measures the percentage of correctly classified images. We considered the dataset shifts that may occur due to variations in image quality, lighting conditions, and other environmental factors. To mitigate these effects, we applied various data augmentation techniques to the training set. Our experiments also revealed that data augmentation techniques such as random resize crop, horizontal flip, vertical flip, shift scale rotate, and cutout were highly effective at improving model performance. By applying different augmentation techniques, we are expecting to increase our model robustness and make it have better performance in prediction with unseen dataset. We also found that certain loss functions, optimizers, and learning rate schedulers were more effective than others, and that the performance of some models was highly sensitive to the choice of model settings and hyperparameters. Our results show that on a 30% held-out testset, advanced models like EfficientNet and Vision Transformers outperform traditional models like VGG19 and ResNet, achieving accuracy of up to 88.26%. We also found that using data augmentation techniques and optimizing hyperparameters, loss functions, and optimizers can significantly improve the accuracy of all models. However, when choosing data augmentation techniques, we need to be careful as improper choice may decrease the accuracy.

By submitting our model to the kaggle leaderboard, we find out that generally a high test accuracy will result in a high LB score, as shown in the table. However, some traditional models have less robustness in predicting for unseen images, which results in a slight decrease in LB score. Bilinear-CNN may be too sensitive to noise so it has a lower score. Overall, our advanced models can have a better LB score which is about 87-88. Besides, we find that larger image size can result in a better accuracy and score but it needs longer training time. So if we are able to train with 512*512 images, we may expected that EfficientNet and resnext model can get a even better result. Overall, our experiments demonstrate that high accuracy in classifying Cassava Leaf Diseases can be achieved using a variety of machine learning models, and that efficient models like EfficientNet and Vision Transformers can achieve high accuracy with less computational resources.

In the last, we tried several combinations of individuale model. We find that by using ensemble, we can still improve the leaderboard score by 0.5. Therefore, that's the reason why all the top solutions are using ensembles of different kinds of models. By combining those top individual models, we can analysis there performance by confusion matrix, find out which part of dataset is being predicted well, and then form a proper weights for the models.

6) Compute/Other Resources Used

For training process, we mainly working through AWS sagemaker and save the best weights. We use Kaggle Notebook for model analysis, model inference and submitting our results to the LeaderBoard. We also borrow several code from Kaggle about the implementation for different

kinds of loss functions in Pytorch. Besides, several ideas are inspired from Kaggle competition about the possible choice of loss function, optimizer, augmentation and LR scheduler.

7) Conclusions

Outcomes: We find that in general advanced model can have a better performance but need more computational resource. For Kaggle leaderboard, we find that generally, high test accuracy will direct mean high LB score, but for some models, it is sensitive to noise or it has lower robustness, which result in a lower LB score. Robustness is also closely related to the augmentation we chosen. By add different kinds of augmentation, we hope to make the model be able to learn the complex image data, but we still need to make sure that the augmentation method should be carefully chosen as if we add too much augmentation, the model will not be able to learn such things. In the end, we find that ensemble can perform better than individual models and result in a higher leaderboard. The weights of different model also matter a lot. Average is the easiest choice but a carefully designed weight, like weighted average can have better result. For our case, we get the best LB score by use a weighted average of the top 3 individual models. The weights are chosen based on the top 3 percent of mispredicted labels from the confusion matrix.

In Hindsight: We find out that if we train our model from scratch and don't use the pretrained model, it will be hard to get a good accuracy. Therefore, we decided to keep using the pretrained weights. Besides, we find that it will take a really long time to train larger models, so it requires us to pay more attention to the choice of augmentation methods to avoid overfitting and trying different loss functions to make it converge fast.

Which parts of your original proposal plan were you unable to execute? Mainly we did everything we mentioned in the proposal. There are two optional choices in our proposal. We don't have time to try NFNET. For introducing computer vision preprocessing methods, we find it will slow down the training process so we decide not to do so. By contacting our project mentor TA, we know more about how different models work and get the idea about how to choose appropriate augmentation methods and build good ensembles.

Ethical Considerations, and Broader Social and Environmental Impact:

Our project aims to address the issue of accurately identifying cassava leaf disease, which can improve crop yield and food security in areas where cassava is a staple crop. By providing an efficient and reliable method for disease identification, our project can potentially reduce the use of pesticides, which can have positive environmental impacts by minimizing their negative effects on soil health and water quality. However, it is important to consider potential ethical implications and broader social impacts of this technology. If our method becomes the primary tool for disease identification, it could lead to decreased reliance on traditional methods of crop management and human expertise. It is important to consider strategies for ensuring equal access to the technology, while also preserving traditional knowledge and expertise in cassava cultivation.

8) Roles of team members:

Zesheng Liu: Build the pipeline of preprocessing and evaluation, train Vit, BCNN and ResNext, submit models to Kaggle and try ensembles, try different augmentation and loss function

Xinxin Liu: Train baseline models, CNN, ResNet, and VGG. Try random crop and color augmentations.

Ruilai Xu: Trained various approaches on different EfficientNet models. Also, tried different augmentations on those models to get better results. Conducted paperworks and general results.

(Exempted from page limit) Other Prior Work / References (apart from Sec 3) that are cited in the text:

1. Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Visual Recognition", ICLR 2015
2. K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," CVPR 2016
3. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021
4. Mingxing Tan, Quoc Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", PMLR 2019
5. Xie, Saining & Girshick, Ross & Dollar, Piotr & Tu, Z. & He, Kaiming, "Aggregated Residual Transformations for Deep Neural Networks", CVPR 2017
6. <https://www.kaggle.com/code/abhinand05/vision-transformer-vit-tutorial-baseline>
7. <https://www.kaggle.com/code/khyeh0719/pytorch-efficientnet-baseline-train-amp-aug>
8. <https://www.kaggle.com/code/piantic/no-tta-cassava-resnext50-32x4d-inference-lb0-903/notebook>
9. <https://www.kaggle.com/code/piantic/train-cassava-starter-using-various-loss-funcs/notebook>
10. T. -Y. Lin, A. RoyChowdhury and S. Maji, "Bilinear CNN Models for Fine-Grained Visual Recognition," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1449-1457, doi: 10.1109/ICCV.2015.170.

(Exempted from page limit) Attach your midway report here, as a series of screenshots from Gradescope, starting with a screenshot of your main evaluation tab, and then screenshots of each page, including pdf comments. This is similar to how you were required to attach screenshots of the proposal in your midway report.

Project Midway Report (1 submission per group, add all members in your group)

● Graded

Group

Xinxin Liu

Zesheng Liu

Ruilai Xu

[View or edit group](#)

Total Points

7 / 7 pts

Question 1

Evaluation Question [select all pages]

7 / 7 pts

✓ + 1 pt Does the report follow the provided template including the 4-page limit (excluding exempted portions), with reasonable responses to all questions?

✓ + 2 pts Has feedback from the last round been effectively addressed?

✓ + 1 pt Has the team identified a clear topic and viable new target contribution, as per the project specifications provided in class?

✓ + 1 pt Has the team moved in a non-trivial way towards their target contribution?

✓ + 2 pts Has a clear and systematic work plan been formulated for the remaining weeks?

Great job! As you have built up infrastructure like data processing, data augmentation, as well as the baseline models + bilinear CNN, it seems you've made good progress. One note is to consider looking closely at the loss curves for the different models to try to parse if the training can be improved (with different lr, weight decay, lr warmup, etc).

Fine Grained Image Classification

Team: Zesheng Liu, Xinxin Liu, Ruilai Xu. **Project Mentor TA:** Anish Bhattacharya

1) Introduction

Set up the problem: Cassava, a major staple food crop in Africa, is affected by various leaf diseases that can lead to significant yield losses. In this study, we aim to classify Cassava Leaf Diseases using a dataset of labeled images provided by the Cassava Leaf Disease Classification Challenge. The dataset consists of 21,397 images, each containing a single leaf and the corresponding disease label, with five possible classes. Our machine learning model will take an input image and output the corresponding disease class.

In our project, we will focus on classifying the disease label for those leaf images. Our input will be a combination of image data. Our raw images are in RGB format and the size is 600×800. Our model output will be probability values representing the probability of a single leaf belonging to different classes.

Implementation: We will implement different kinds of models. Our baseline model will be multi-layer CNN. Then we will come up with the traditional VGG and ResNet architectures. Finally we will implement advanced models like Bilinear CNN, EfficientNet and Vision transformer. In implementation, we will try different loss functions and optimizers. We will keep the image size and batch size to be constant for all the models.

Evaluation: In our evaluation, the main factor will be the accuracy of prediction, which will be evaluated either on a held-out set from the whole given dataset or through the Kaggle Leaderboard. We will try different image augmentation methods and evaluate their influence on final accuracy. We may also try to evaluate the effect of test time augmentation and different combinations of ensembles.

2) How We Have Addressed Feedback From the Proposal Evaluations

-Decision about whether to train from scratch or use pretrained weights:

After talking with our project mentor, we confirm that for our baseline model, we will train from scratch. For VGG and ResNet, we will try to do both training from scratch and use pre-trained weights for fine tuning. If we really have trouble in training from scratch, i.e. having an extremely low accuracy, then we will only start with pretrained weights. For those advanced models, we will use pre-trained weights and do fine tuning based on these weights.

-Clarify about 'implement these models' in the proposal: We will mainly write our own code for different models, but we will use some existing packages to easily build our model architectures.

-Data pre-processing pipeline: This will be something we plan to try in the final period. We hope to introduce some image preprocessing package for denoising and some other network for attaining a bounding box and crop the image based on the bounding box.

3) Prior Work We are Closely Building From

- A. <https://www.kaggle.com/competitions/cassava-leaf-disease-classification/discussion/207450>
A few steps for starting, generally give out some idea about what we can explore for this dataset.
- B. *Fine-Grained Image Analysis with Deep Learning: A Survey*
<https://arxiv.org/abs/2111.06119> A detailed review paper of fine grained image classification

4) Contributions

4.1 Implementation Contributions

We plan to implement several different models and compare their performance. Our baseline model would be multi-layer CNN. Through the implementation and analysis of baseline model results, we hope to get more familiar with our dataset and try to gain some idea about the expected model capacity we need for a high accuracy. Also, we will add the batchnorm and dropout, aiming to find out whether there would be a possible overfit situation.

Then, we will implement the most famous VGG and ResNet architecture. These models are widely used for image classification and feature extraction. By implementing these models, we will expect to have a higher accuracy compared to our baseline model but may be lower than those advanced models. Some advanced models like Bilinear CNN and EfficientNet are also built on those traditional VGG or ResNet architectures, either directly use them or use the idea of convolution blocks and skip connection.

For our advanced models, we plan to implement these three models: Bilinear CNN, a famous architecture for fine-grained classification; EfficientNet and Vision Transformer, models that have top accuracy on image classification. We will use pretrained weights and some existing packages(i.e. timm for pytorch) to easily build the model and then do fine tuning on them. At last, we will do a brief exploration of ensembles, trying to combine different models together.

While implement different models, we plan to consider a few essential issues:

-For the choice of loss functions, we will start with the cross entropy loss to train our model. After making everything working properly, we will consider changing to other loss functions like bi-tempered loss, focal cosine loss, focal loss, label smoothing, etc.

-For the optimizer, we will consider choosing from Adam or SGD with a proper scheduler for weight decay. Choice of the optimizer, corresponding hyperparameters and scheduler can be different for different models but common choice(i.e. lr=1e-3/1e-4) would work and this will not have a big influence on accuracy.

-For batch size and image size, we will start it with batchsize=128 and image size between 200*200 to 400*400 at first. When we train all our models in the end for comparison, we will consider the training time and memory needed for different models and decide a common choice for all the models.

For image pre-processing, we hope to build a whole pipeline, including resize/random crop, possible augmentation methods that found to be useful for performance, and denoise. We may also try to use other network for locate the feature and crop the image instead of random crop, if time allowed.

4.2 Evaluation Contribution

Our task will be aiming for a high accuracy. This will be the main factor for evaluating our model. We plan to either evaluate the accuracy both on a held-out test dataset from the given whole dataset and by the Kaggle Leaderboard score(especially the private board). We plan to consider the effect for different augmentation methods(Normalize, Randomcrop, Resize, Flip, Cutmix...) for pre-processing the dataset. Depending on our process, if we have enough time in the end, we will consider exploring the effect of TTA(test time augmentation), which will apply different transformations to test images and add them back to the test set, and different methods to combine models in ensemble.

4.3 Experiments and Results

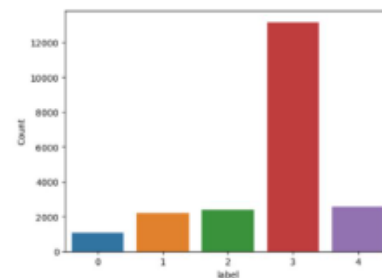
(1) Feature Engineering

First we did basic feature engineering and exploration of the dataset. There are mainly two things that need to be noticed:

- **Data imbalance:** It is clear that class 3, "Cassava Mosaic Disease (CMD)", is the majority, which accounts for more than half of the whole dataset. An imbalanced dataset may cause the model to learn mostly from the majority class. Therefore, in order to get a high accuracy, when we finetune our model hyperparameters, we need to resolve this problem. We may consider converting the classification process into two

processes: First we do a binary classification to classify whether it belongs to class 3 or other classes. Then, we do another multi-class classification for the remaining 4 kinds of classes.

- **Image size and Noise:** The original image size is 600×800 pixels, which is quite big. In order for a higher training speed, we will reduce the image size by either directly resizing the original image or do random crop. Besides, we notice that raw image have some background noise, so random crop could be helpful



(2) Baseline models

For our baseline model, we build a multi-layers CNN model and train the model with a subset of the original class, to get some idea about the model capacity we need and whether it will easily overfit the model. Dataset for two baseline models are subsets from the 21,367-image package. Training set is the 0~2047 th images (2048 in subtotal), and the validation set is 2048~2304 th images (256 in subtotal).

For our first implementation, we propose a baseline multi-layer CNN model, following the architecture: 6 times {Convolutional layer; Max-pooling layer;ReLU activation }; Fully connected layer; Output layer with Softmax activation. The validation accuracy achieves 63.28%

The second implementation extends the baseline CNN model by adding Batch Normalization [1] and Dropout [2] techniques to improve the model's generalization ability and reduce overfitting. We add batch normalization and dropout after 3 convolution blocks.

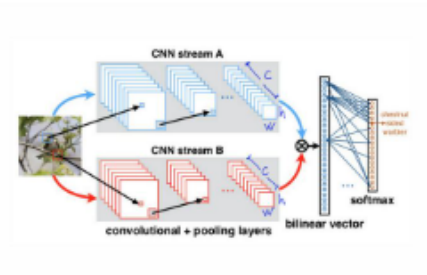
The validation accuracy is 10.16%, which is much lower than the very basic 6-layered CNN model. The bad performance illustrates that the original baseline CNN model is not overfitted so that batchnorm & dropout is redundant and useless.

We plan to vary the number of filters, kernel size, and the dropout rate to analyze the impact of these hyperparameters on the baseline model's performance, and may try to retrain the model after addressing the data imbalance problem.

(3) Traditional architecture and advanced model

Here we mainly report the implementation of bilinear CNN, which is a model for fine-grained image classification.[3]

The architecture is shown in the figure. In the model, we have two individual feature extraction parts, which is CNN stream A and CNN stream B. Here we can change CNN to any other feature extraction architecture. One understanding is that in stream A the network will locate the feature position in the whole image, while in stream B the network does feature extraction. Then, we do a bilinear



pooling process to combine the output of two streams, where we do a matrix multiplication for features in the same feature map location and sum up among all the channels. After summing up among different channels, we convert it to a bilinear vector and normalize it. Then in the end we can use a fully connected layer and softmax for classification.

In our implementation, we set the stream A and B to be the same ResNet 34 with pretrained weights. We use the batch size to be 128, input image size 256*256, Adam optimizer with learning rate 1e-4 and weight decay 1e-6, and a Reduce LR On Plateau scheduler. We take 30 % of the whole dataset to be a test set. We train both the parameters for the ResNet and the fully connected layer for 50 epochs. The highest accuracy on test dataset is 83.69%

We also try to run the EfficientNet B3 model[4] for the same hyperparameter setting and get a accuracy of 83.8%.

5) Risk Mitigation Plan

We have already implemented the code for reading the dataset, doing augmentation through a certain package and building the Bilinear-CNN model in pytorch. This can serve as a guideline for building other models by only changing the way how we define the model and using the timm package. The accuracy of different models agrees with our expectation. For the remaining time, we will first set up individual notebook for training EfficientNet and ViT, and get some idea about the estimate training time. ViT can take longer time but no longer than half day. In implementation different models and try different loss function and hyperparameter setting, we should have no risk but just keep the code running.

Possible risk can be within the evaluation part, especially with the Kaggle leaderboard, as we don't know the size of the private testset. If the inference takes a really long time for the whole testset (with TTA then it will need even longer as we increase data), then we plan to do evaluation only based on held-out set(~5000 images). Another risk is that as the accuracy are really close(we get ~83% now and the highest is 91%), it may be hard to evaluate the difference between different model. If we failed to evaluate only based on accuracy, we will try to analysis the reason(possible connect with the noise and mislabeled problem) and try to choose other metrics(F1 score, Mathews Correlation Coefficient, Categorical log loss, ROC-AUC etc) to better evaluate the model performance.

(Exempted from page limit) Other Prior Work / References (apart from Sec 3) that are cited in the text:

- [1] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift Sergey Ioffe, Christian Szegedy Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:448-456, 2015.
- [2] Srivastava, Nitish & Hinton, Geoffrey & Krizhevsky, Alex & Sutskever, Ilya & Salakhutdinov, Ruslan. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929-1958.
- [3] T. -Y. Lin, A. RoyChowdhury and S. Maji, "Bilinear CNN Models for Fine-Grained Visual Recognition," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1449-1457, doi: 10.1109/ICCV.2015.170.
- [4] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019. Arxiv link: <https://arxiv.org/abs/1905.11946>.

(Exempted from page limit) Attach your proposal here, as a series of screenshots from Gradescope, starting with a screenshot of your main evaluation tab, and then screenshots of each page, including pdf comments. For example:

Project Proposal (1

● Graded

submission per group, add all members in your group)

Group

Xinxin Liu
Ruilai Xu
Zesheng Liu
[View or edit group](#)

Total Points

3 / 3 pts

Question 1

Grading Question [Please mark all pages for this] 3 / 3 pts

✓ + 1 pt

Identification of clear target contribution for Parts 1 & 2. For Part 1, the contribution should involve a substantial implementation component (e.g., not just trying a bunch of classifiers implemented in sklearn). Examples include significant feature engineering (beyond polynomial features, e.g., bigrams/word preprocessing for NLP tasks) or architecture engineering (e.g., try a new CNN architecture). For Part 2, there should be clear hyperparameters that are tested, as well as a clear distribution shift to be tested.

✓ + 1 pt

Listing relevant prior work. There should be at least one paper that is closely related to the topic (e.g., not a generic CNN citation).

✓ + 1 pt

Followed template, with reasonable responses to remaining questions.

+ 0 pts

(TAs: A set of things to consider when adding free-form comments)

- Is this project feasible given the time & compute limitations, and team expertise?
- Have they left out related work you are aware of?
- Is the risk mitigation plan good enough, does it aim to build up a minimum viable product quickly?
- Are there ethical concerns?

Hi Zesheng,

Thanks for reaching out. You discuss a lot of different methods, which is good, but since you have not decided on using pre-trained models or training from scratch, you risk running out of time. I would make those decisions as soon as you can. You may need time to tune the models to see if training from scratch is tractable or not, so please account for that.

I am a bit confused about what you will be implementing. You say you will "implement these models"; so to clarify, you are writing these from scratch? I see you will probably build a pre-processing pipeline, which is a good implementation contribution.

You included good references, and it seems like your group understands the project well.

I would keep in mind that whatever results you get, discussing it and providing explanations should get you most of the points. Even if your method(s) may not work, you can get full credit in the project if you show you implemented & evaluated enough.

Anish

Title: Fine Grained Image Classification

Team: Zesheng Liu (CIS 5190), Xinxin Liu (CIS 5190), Ruilai Xu (CIS 5190)

Kaggle dataset: Cassava Leaf Disease Classification

Part 1: Implementation contributions: We consider to implement these models: 1) Baseline multi-layers CNN model; 2) Baseline model+Batchnorm and Dropout; 3) VGG model; 4) ResNet; 5) EfficientNet; 6) Vision Transformer; 7) Ensembles.

For model 1 and 2, we will train it from scratch. For model 3 and 4, we plan to try to train it from scratch, if we found it would take too long time then we may use pretrained weights. For model 5 and 6, we will use pretrained model and fine-tune. For Ensembles, we may try different weights instead of simply majority vote or average the prediction probability. We plan to use the timm(<https://github.com/huggingface/pytorch-image-models>) package for those pretrained models. We plan to use PyTorch as the main framework for those models that we will train from scratch.

If time allows, we will consider to do the following: 1) Refer to several Kaggle discussions, the dataset may have noise then we may add a preprocessing model for denoise. 2) We may use the OpenCV package and implement several computer vision techniques for preprocessing. 3) We may implement some methods that are designed for only fine grained image classification, like the new SOTA model NFNets.

Part 2: Evaluation contributions: We will evaluate the model performance by a held-out test dataset and by the Kaggle Leaderboard score which accounts for the accuracy of our prediction. For each architecture, we plan to focus on these aspects: 1) The chosen of image size 2) What kind of data augmentation methods are used, as this may be the most important part for a high accuracy 3) Which loss function is used, as we may use other loss function instead of cross entropy loss and we may include label smoothing 4) We will consider the effect of TTA(test time augmentation) which will apply different transformations to test images and add them back to test set. 5) How different models in the ensemble are being combined. Usually we use majority vote but it may not be the case for our dataset.

Prior work:

1. <https://www.kaggle.com/competitions/cassava-leaf-disease-classification/discussion/221957> Kaggle 1st Solution
2. <https://www.kaggle.com/competitions/cassava-leaf-disease-classification/discussion/207450> A few steps for starting
3. <https://www.kaggle.com/competitions/cassava-leaf-disease-classification/discussion/221113> 16th place solution with detailed and readable code for beginner
4. <https://arxiv.org/abs/2111.06119> A review paper of fine grained image classification

Which parts of the curriculum from this class do you expect to apply?: We plan to use deep neural networks such as CNN, especially VGG architecture that are suitable for image data, since we are using image data and we have a sufficiently large dataset for deep learning.

Besides, we will use the concept of ensemble to combine different model together for prediction. We will also use the idea of transformer and self-attention for vision classification.

Expected challenges and risk mitigation:

1. Data Set size and quality
The site provides 21418 images in different types and classes, different lighting conditions and different photo scales, which may lead to inequality and ambiguity. The preprocessing procedures (like normalization methods) should be careful considering.
2. Limitation of computational resources and runtime
The whole data package is about 6GB, which is an intermediate size. We are still not sure the given storage and GPU is sufficient for training more complicated nn models. Besides, SOTA classification model like ViT will need a really large dataset to train from scratch, even fine-tune of pretrained weight may take a considerable time.
3. Generalizability and Robustness
In the real world, the scene we meet is much more thorny than what we trained by input images. We should think carefully about balancing training score and actual performance when facing unexpected kinds of leaves. In addition, maybe some features or interference not appearing in our training set should also be taken into consideration, so that the way we add noises is also important. We may finally find that our method with the highest accuracy may not work for another fine grained classification task.
4. The choice of criteria evaluating model's performance
Avoiding overfitting highly improves the performance in the real world. Also, we should think of whether TPR or simple accuracy is more attentive to the needs. Therefore, we need carefully chosen the loss function and the hyperparameters. Besides, for our dataset, we need to carefully choose the best method for training data augmentation and TTA.

Ethical considerations and broader social impact:

Good:

1. Accurately identifying and treating cassava leaf disease can improve crop yield, which could help alleviate hunger and improve food security in areas where cassava is a staple crop.
2. If the disease can be accurately identified, farmers may be able to reduce their use of pesticides, which could have positive environmental impacts.

Bad:

1. If the method becomes the primary tool for disease identification, it could lead to decreased reliance on traditional methods of crop management and human expertise. This could lead to a loss of knowledge and skills.
2. If this method is only accessible to certain farmers or organizations, it could exacerbate existing inequalities and create economic barriers to accessing technology that could improve crop yield. It is important to consider how to ensure equal access to the technology and ensure that it does not perpetuate existing inequalities.

(Exempted from page limit) Supplementary Materials if any (but not guaranteed to be considered during evaluation):

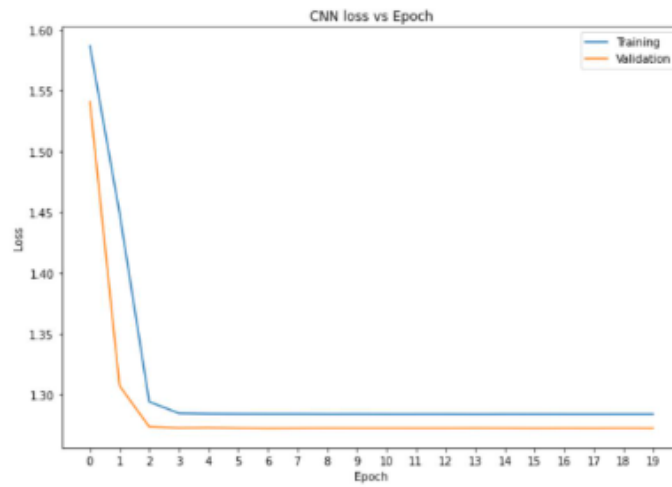


Figure 1 Loss versus epoch for baseline CNN model

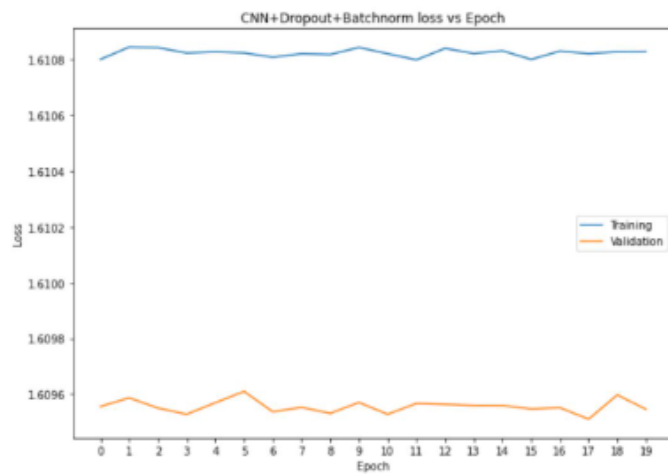


Figure 2 Loss versus epoch for baseline CNN model+batchnorm+dropout

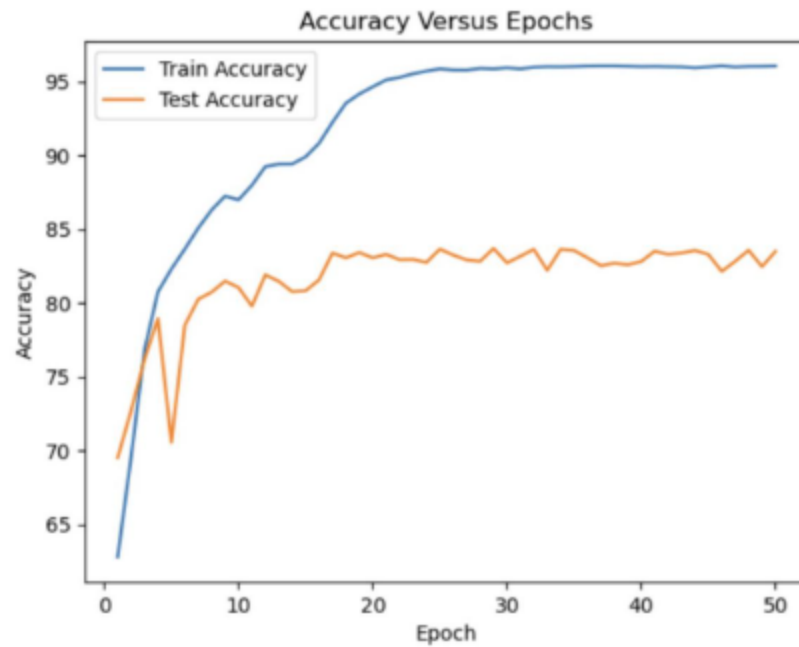


Figure 3 Accuracy versus epochs for Bilinear CNN

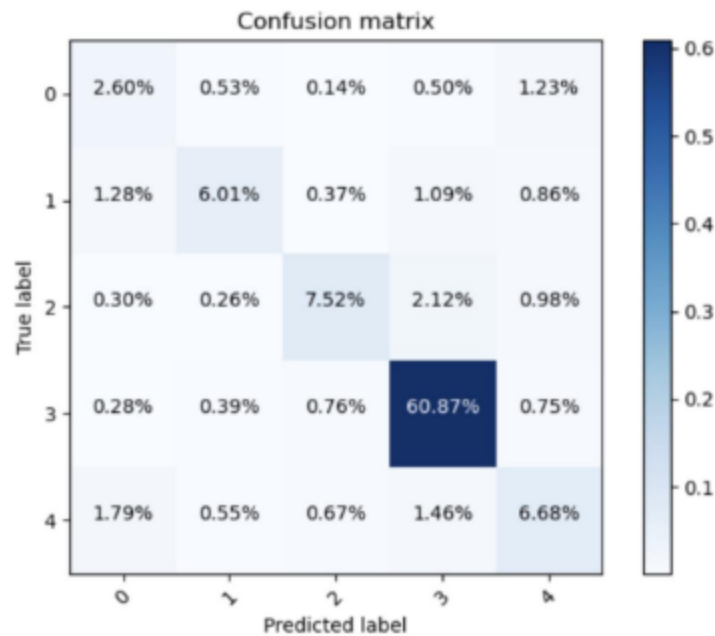


Figure 4 Confusion matrix for Bilinear CNN















Code File for Baseline Model: <https://www.kaggle.com/zeshengliu/baselinemodel>

Code File for BCNN: <https://www.kaggle.com/zeshengliu/traditional-bcnn>

(Exempted from page limit) Supplementary Materials if any (but not guaranteed to be considered during evaluation):

GitHub Link for code: <https://github.com/StarLiu714/Cassava-Fine-Grained-Image-Classification>

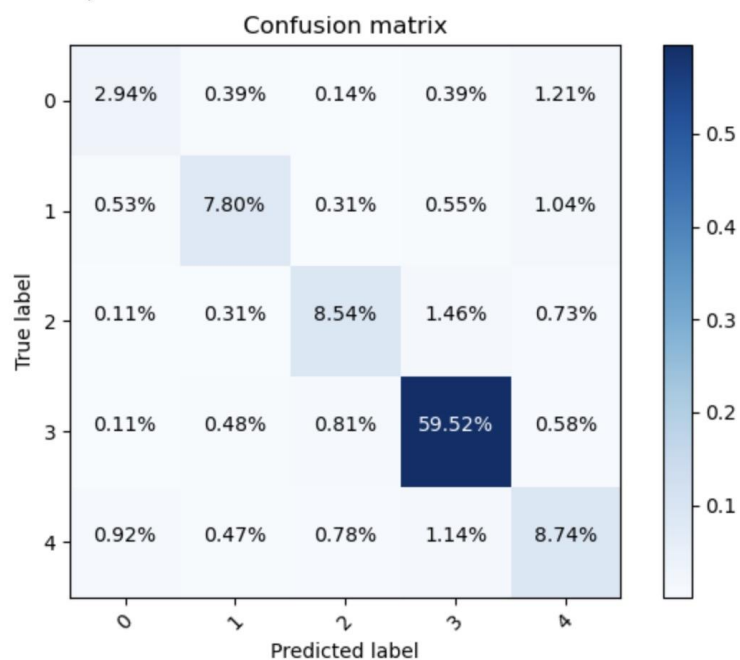
LB Results:

	Fork of Model_Inference - Version 39 Succeeded (after deadline) · 9m ago · ResNet34d_resize	0.7858	0.7793	<input type="checkbox"/>
	Fork of Model_Inference - Version 38 Succeeded (after deadline) · 10m ago · ResNet50d_resize	0.7567	0.7621	<input type="checkbox"/>
	Fork of Model_Inference - Version 37 Succeeded (after deadline) · 10m ago · VGG_resize	0.7801	0.7858	<input type="checkbox"/>
	Fork of Model_Inference - Version 43 Succeeded (after deadline) · 5m ago · B4-86.32	0.8559	0.8559	<input type="checkbox"/>
	Fork of Model_Inference - Version 42 Succeeded (after deadline) · 5m ago · B4-85.14	0.8256	0.8351	<input type="checkbox"/>
	Fork of Model_Inference - Version 41 Succeeded (after deadline) · 10m ago · B3-87.94	0.8771	0.8824	<input type="checkbox"/>
	Fork of Model_Inference - Version 40 Succeeded (after deadline) · 10m ago · B3_87.66	0.8649	0.8661	<input type="checkbox"/>
	Fork of Model_Inference - Version 46 Succeeded (after deadline) · 5m ago · ResNext_87.18	0.868	0.8686	<input type="checkbox"/>
	Fork of Model_Inference - Version 45 Succeeded (after deadline) · 6m ago · ResNext-87.16	0.8473	0.8491	<input type="checkbox"/>
	Fork of Model_Inference - Version 44 Succeeded (after deadline) · 7m ago · BCNN_resize	0.7828	0.7919	<input type="checkbox"/>
	Fork of Model_Inference - Version 49 Succeeded (after deadline) · 20m ago · Vit_8825	0.8742	0.8815	<input type="checkbox"/>
	Fork of Model_Inference - Version 48 Succeeded (after deadline) · 22m ago · vit_8817	0.8836	0.8817	<input type="checkbox"/>
	Fork of Model_Inference - Version 47 Succeeded (after deadline) · 22m ago · VIT_8808	0.8744	0.8815	<input type="checkbox"/>
	Fork of Model_Inference - Version 50 Succeeded (after deadline) · 12m ago · ResNext_87.69	0.8717	0.8699	<input type="checkbox"/>

Confusion Matrix for the top 3 model:

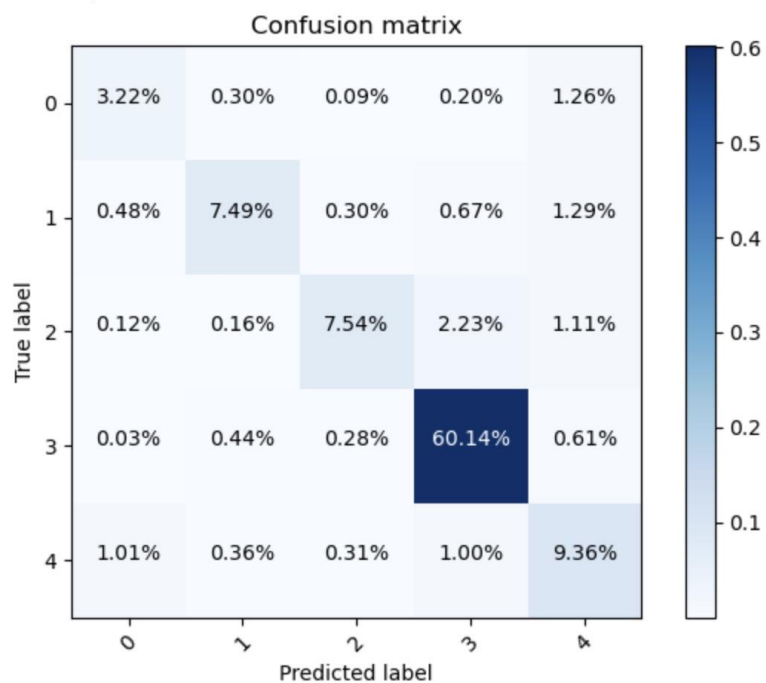
Efficient Net-B3:

Accuracy: 0.87538934



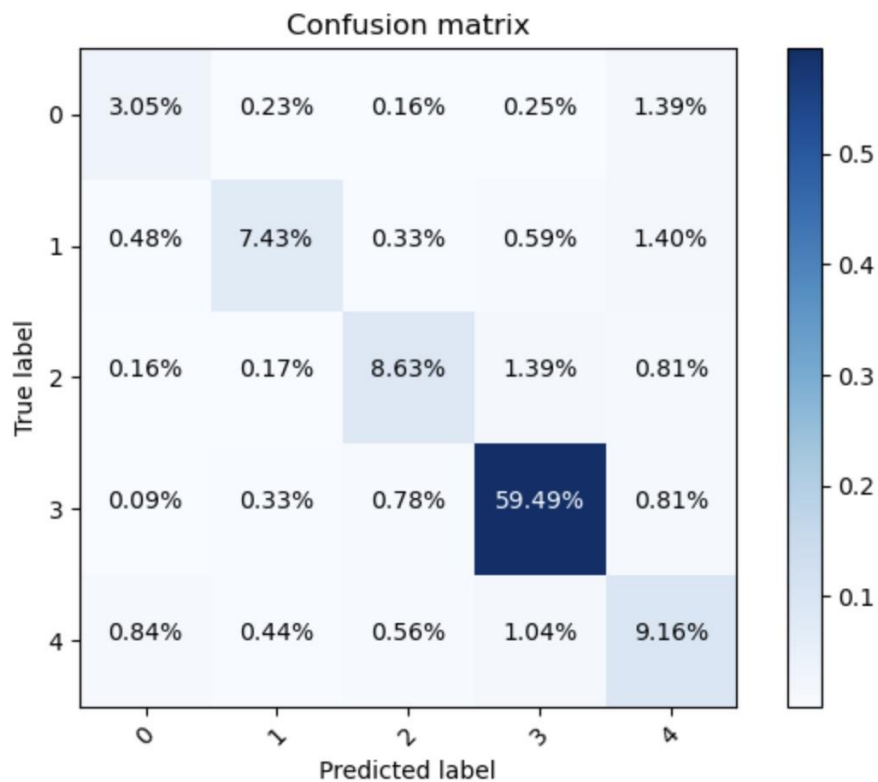
Vit

Accuracy: 0.87777778



ResNext:

Accuracy: 0.9775761



The weights of the best ensemble is:

$y_preds_resnext * (1/1.39 + 1/1.39 + 1/0.84 + 1/3) + y_preds_effnet * (1/1.46 + 1/1.21 + 1/0.92 + 1/3) + y_preds_vit * (1/2.23 + 1/1.26 + 1/1.01 + 1/3)$