Experiments with baseline 2

TLDR- Summary of results:

- added loss and accuracy visualisation
- gradient descent converges
- accuracy after fine tuning : 99.8

Baseline:

We first train the baseline model over 20 epochs.
Structure of baseline model-

---

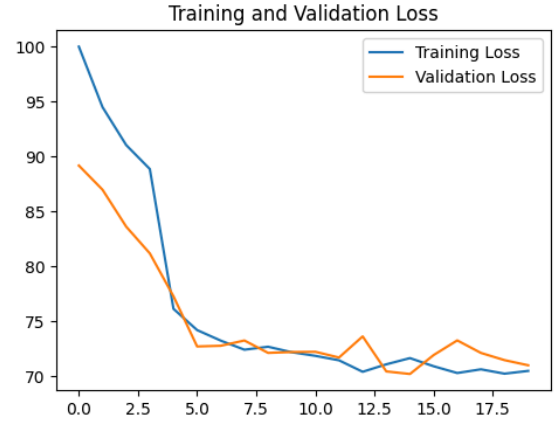
pre trained CNN encoder
Classifier:

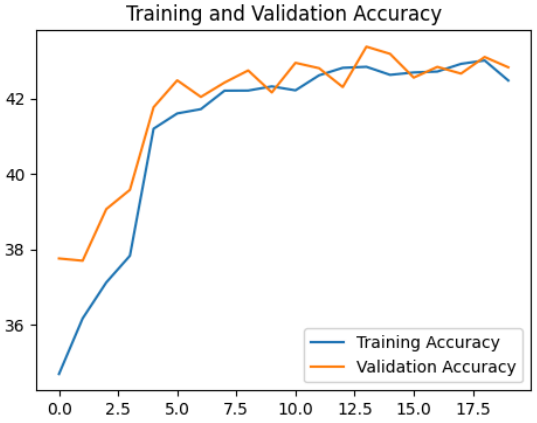
1. dropout layer
2. one fully connected layer with 43 outputs

Before fine-tuning:

```
--------------------
train Loss: 70.4809 Acc: 42.4835
val Loss: 70.9927 Acc: 42.8333

Training complete in 5m 0s
Best val Acc: 43.380952
```

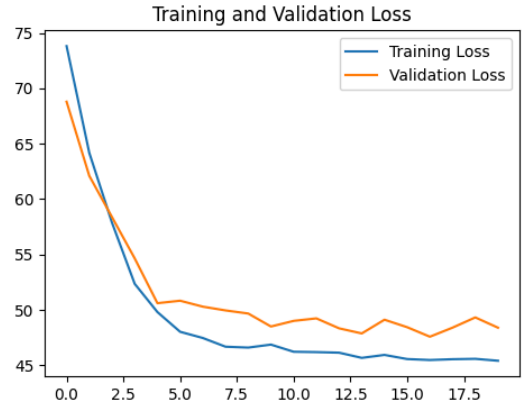


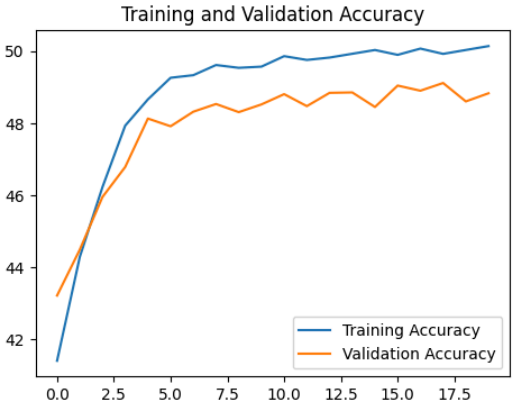


After fine tuning:

```
Epoch 19/19
--------------------
train Loss: 45.4020 Acc: 50.1441
val Loss: 48.3847 Acc: 48.8333

Training complete in 6m 50s
Best val Acc: 49.119048
```





Problem 1: High Bias Given that both training and validation performances are poor, the model likely high bias, and underfitting the data.

Assumption for Problem 1: The model isn't complex enough to capture the underlying patterns in the dataset.

Possible solutions for problem 1:

1. increasing epochs
2. increasing more trainable layers
3. making a more complex classifier
4. removing dropoputs

Problem 2: Gradient descent not converging. I ran the model for 50 epochs at some point, and i can see the loss going down, but really slowly after a few epochs.

Possible solutions for Problem 2:

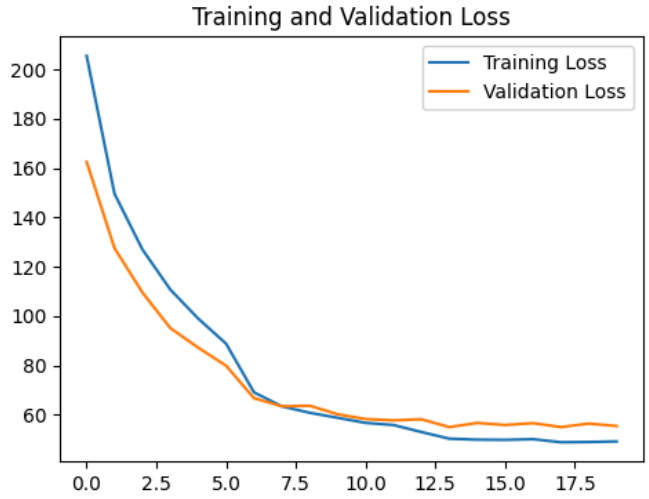
1. removing the schedular and changing alpha manually and monitoring the learning to see what is happening
2. Increasing batch size so SDG converges more smooth and fast
3. normalise data(should do that anyway at some point)
4. If nothing works: use a differnt Optimization Algorithm

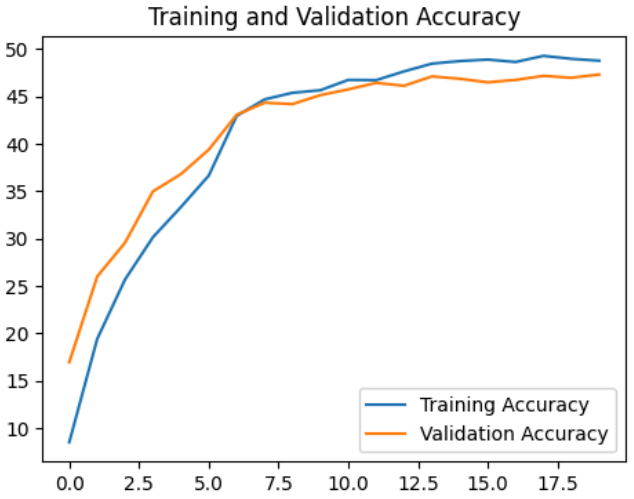
Cool, lets begin

Let us remove dropouts, and make the classifier more dense.

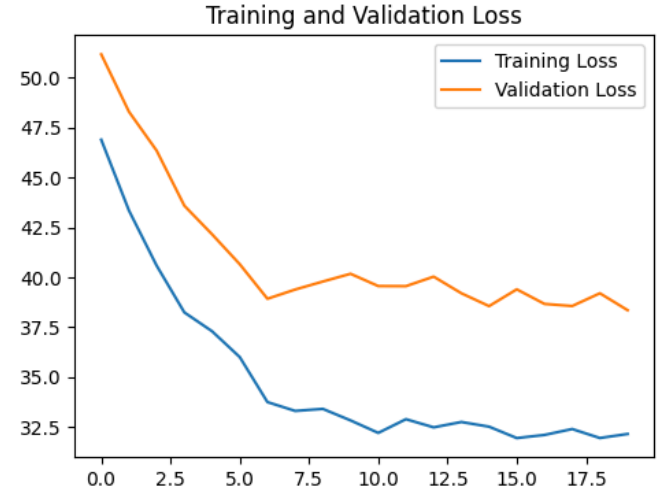
Model now:
Input -> MobileNetV2 (Pre-trained Base) ->
Linear(2048) -> ReLU() ->
Linear(1024) -> ReLU() ->
Linear(512) -> ReLU() ->
Linear(256) -> ReLU() ->
Linear(43) (Output for 43 classes)

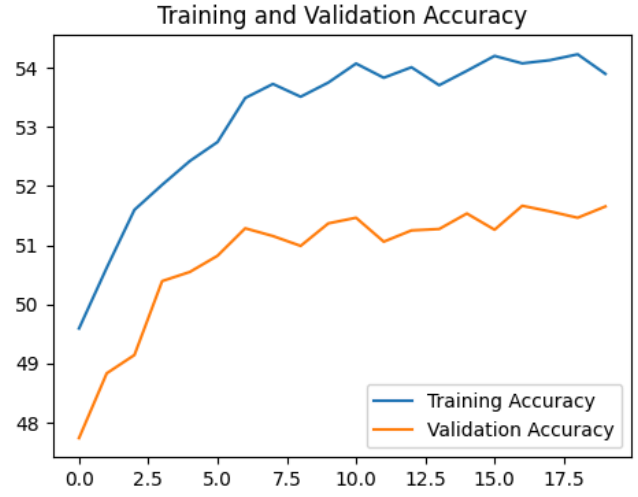
Before finetuning:





After Finetuning



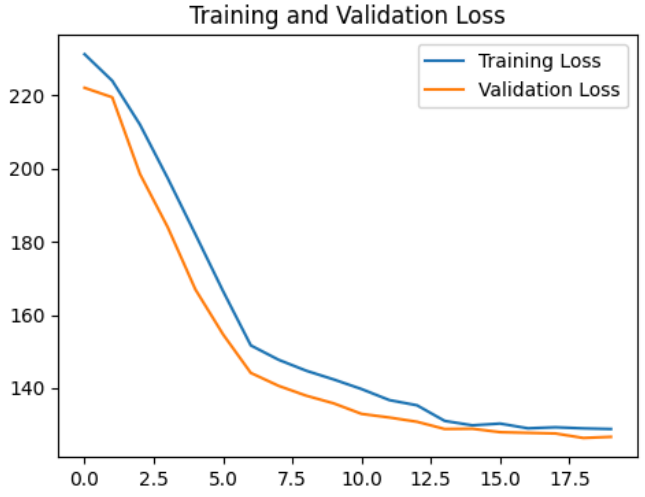


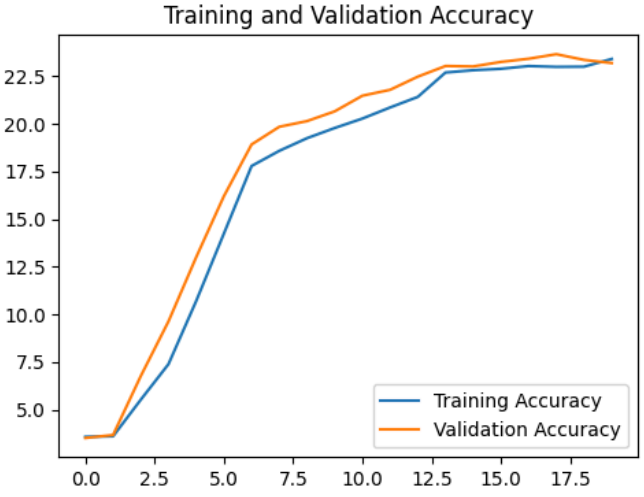
Results: The model is doing better, accuracy is actually increaing.
I played around with different numbers of fully connected layers, and have mentioned the sweet spot which was 5 layers.

exmaple of a bad number fo fully connected layers: Let's see what happens if we make a deeper neural network:
MobileNetV2 Base: Pre-trained feature extractor.
Linear(4096): First dense layer with 4096 units.

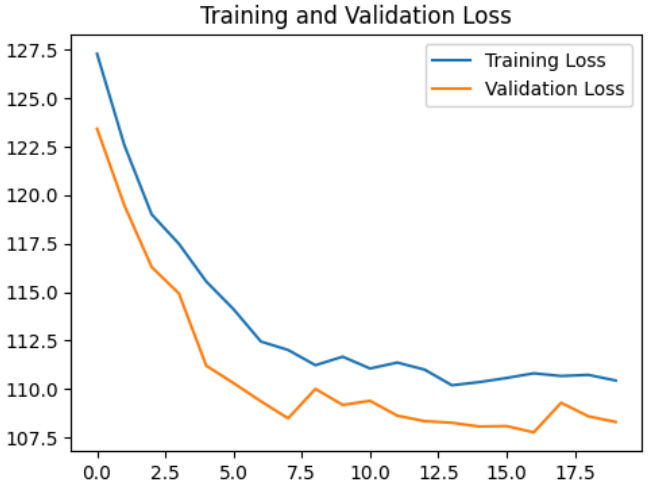
Linear(2048): Second dense layer with 2048 units.
Linear(1024): Third dense layer with 1024 units.
Linear(512): Fourth dense layer with 512 units.
Linear(256): Fifth dense layer with 256 units.
Linear(128): Sixth dense layer with 128 units.
Linear(64): Seventh dense layer with 64 units.
Linear(43): Output layer for 43 classes (GTSRB).

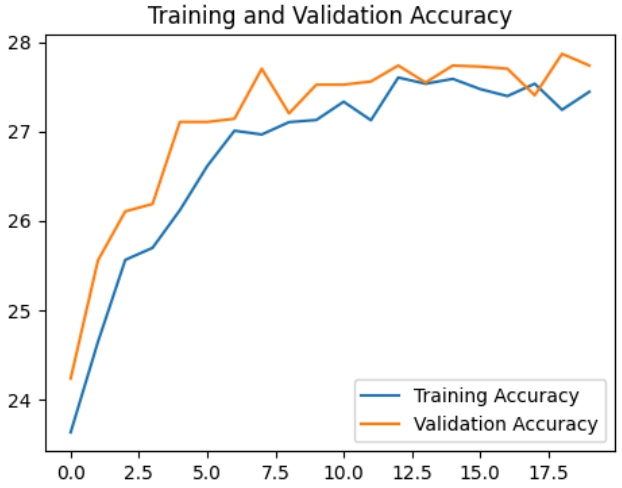
before finetuning:





After finetuning:



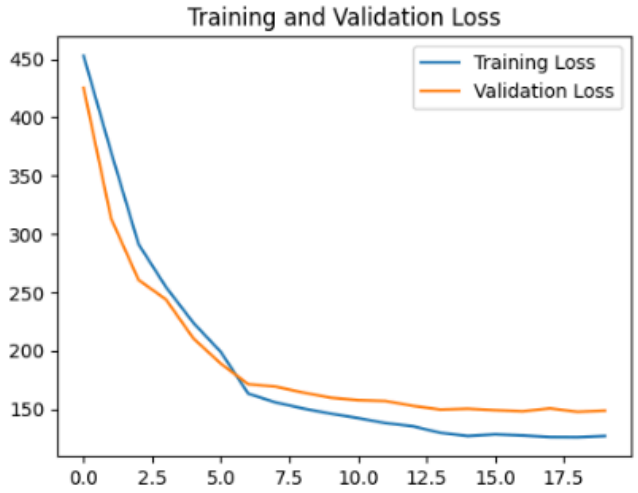


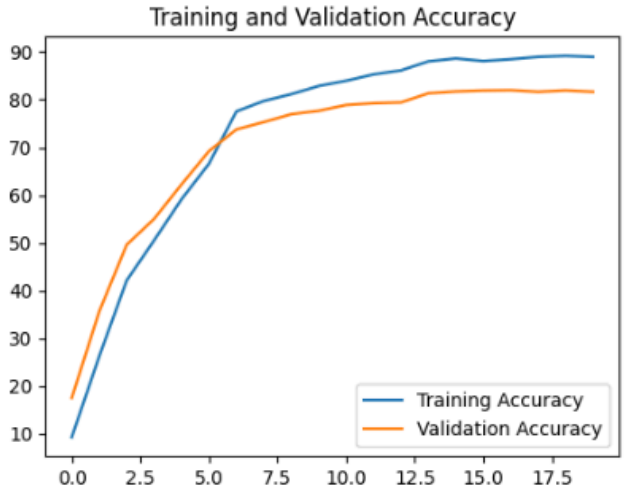
conclusion: That is bad, let us not make it too dense, and stick to 5 layers.

problems at this point:

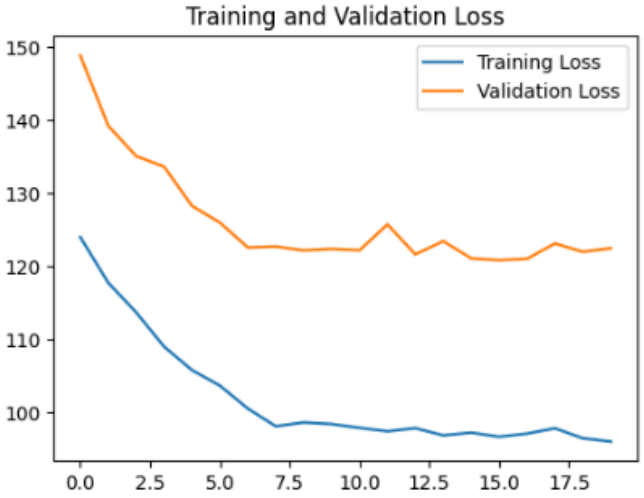
1. learning is wayyyy to slow, to iterate differnt things fast
2. curved are not very smooth

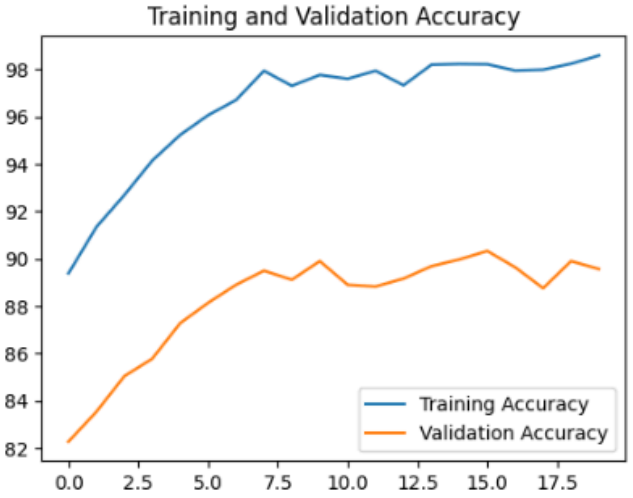
Possible solutions:
increase batch size for mini gradient descent bya power of 2
So new batch size: 2^7
before finetuning





post finetuning





conclusion: this is nice. ML is actually fun?

To do:
pablo wants me to do:

1. Increase the batch size without changinf the architecture.
2. Play with the optimisers, change it.ry diifferent ones.
3. Go through good reports and look them over
4. Read slides of conv net 1 and conv net 2 and this weeks

I want to do:

1. start with the report, non experimental bits. Specifically: Intro, motivation, related work, database description, and evaluation metrics(these things *when done right*, do take time, and not just 1 hour. Also prof told us to start writing)
   1.1 Model architecture, trainign methodoilogy, results and anaylis can be filled in later if we have more things
2. Start formalising measurement metrics with differnt params and hyperparams in a more systematic way. Options: accuracy, F1 score, top K accuracy, confusion matrix. -Should prolly talk to prof. make

this standard across further developmetns by the team

3.(a big maybe) Widening the scope: since our model accuracy is already pretty good now, (i dont really see what we can do to improve it more). So:
Our dataset is very unevenly distriubted across classes. Maybe grouping "similar" classes with very less test images together and redefining the classes with a more uniform like distribution, and running our clasfier there. -should prolly run it by prof