Deep Learning

Assignment 2 Report

**Student Name:** Aniruddha Kalburgi

**Student ID:** R00171450

**Email Contact:** [aniruddha.kalburgi@mycit.ie](mailto:aniruddha.kalburgi@mycit.ie)

**Part A: Convolutional Neural Networks**

**Task (i):**

* Implemented baseline model with 64 filters of size (5, 5), followed by MaxPooling layer of size (2, 2) followed by Flattening and Softmax layer.
* This yielded approximately 40% accuracy, for 70 epochs.
* Although, validation accuracy was pretty much around 40% at 21st epoch where validation loss started increasing above 2.0
* Different ranges of filter sizes like 64, 128, 256 and 512 had almost no effect on validation loss and validation accuracy.
* Even different dropout rates only merely smoothened the validation loss curves, but, no major change.
* Hence, above experiments suggeset that there’s a dearth of enough variety in the training data.

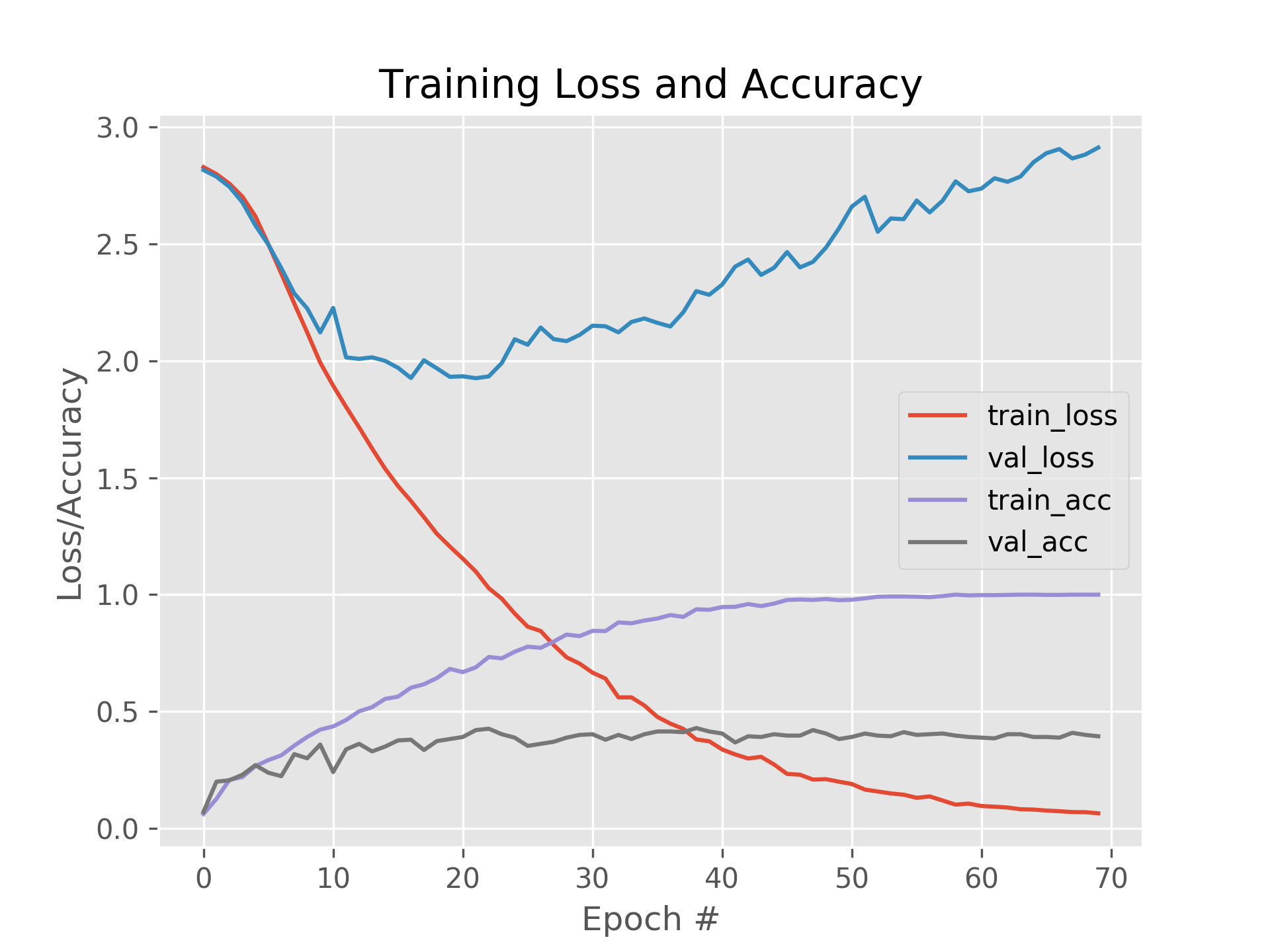


Fig 1: Baseline Model, approx 40% accuracy, 64 filters

**Different Configurations:**

I tried following 3 different configs other than the default one:

**Config 1)**

Conv2d, relu, with different number of filters of size (5, 5) ->

MaxPooling2d layer of size (2, 2) ->

Conv2d, relu, with different number of filters of size (3, 3) ->

MaxPooling2d layer of size (2, 2) ->

Dense, 500 neurons, relu ->

softmax

Along with dropout, with option to enable/disable it.

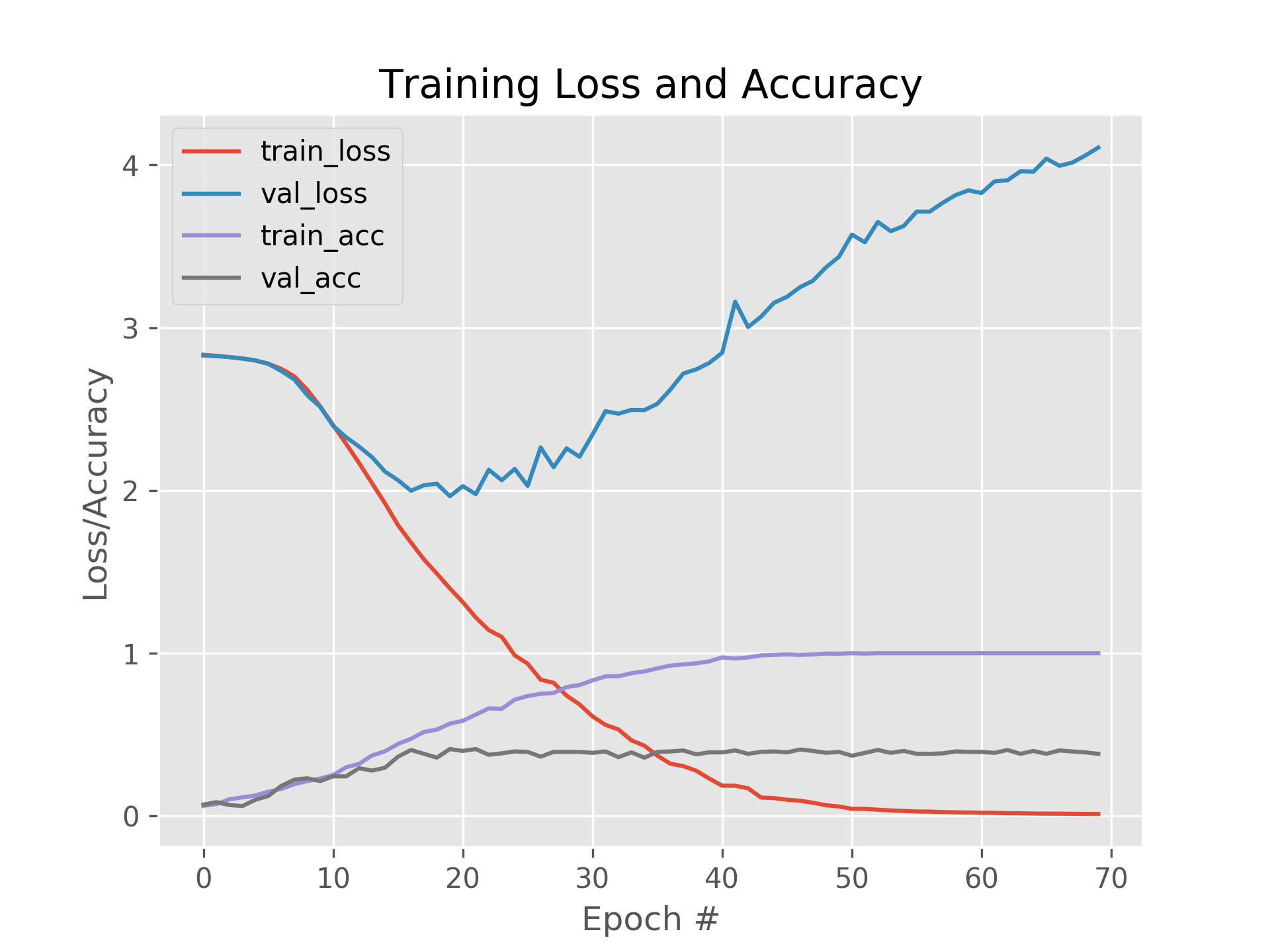


Fig: Config 1, approx accuracy 38%

* Config 1 had slightly poor impact on the validation accuracy, reducing it down approx by 2% than baseline.
* Validation loss increased drastically from 3 to 4+.
* It can also be observed that validation loss in baseline config is below 2 for 20th epoch, whereas in config 1, it has already started going above 2 at 20th epoch.

**Config 2)**

In addition to config 1,

added below layers to form config 2:

Config 1 Conv and Pooling layers ->

Conv2d, relu, with **different number**(explicitly controlled through arguments) of filters of size (3, 3) ->

MaxPooling2d layer of size (2, 2) ->

Config 1 Dense layers ->

Dense, relu, 250 neurons.

Along with dropout, with option to enable/disable it.

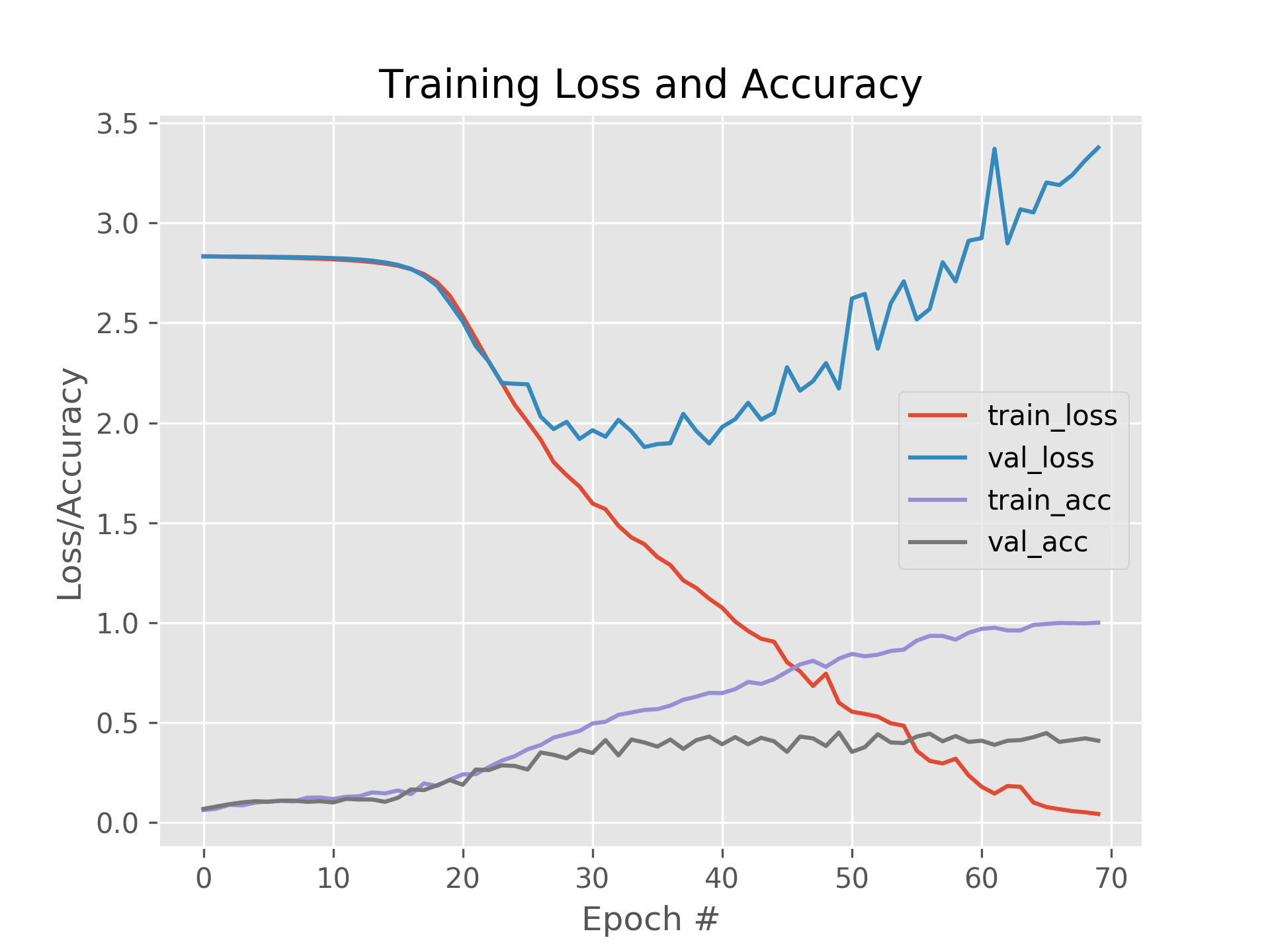


Fig 3: Config 2, approx validation accuracy 45%

* Config 2, showed smoother convergence for the losses and accuracies, where validation loss and accuracies started to diverge after the 21st epoch.
* Yet, config 2 has achieved lower validation loss than config 1 over the 70 epochs.
* The validation accuracy was increased approx by 7% over the config 1, with that reaching pretty close to 50% for some epochs.

**Config 3)**

In addition to config 2,

added below layers to form config 3:

Config 2 Conv and Pooling layers ->

Conv2d, relu, with **different number**(explicitly controlled through arguments) of filters of size (3, 3) ->

MaxPooling2d layer of size (2, 2) ->

Config 2 Dense layers ->

Dense, relu, 100 neurons.

Along with dropout, with option to enable/disable it.



Fig 4: Config 3, approx validation accuracy 48%

* Config 3 showed pretty smoother convergence for all accuracies and losses up to 45 epochs, and then they started to diverge and take noticable erratic curves.
* Yet, the validation loss for 70 epochs was only around 1.7, which is drastic improvement over the previous configs, where the validation losses were above 3.5.
* Even, the validation accuracy, although it started diverging from the training accuracy, led us to 50% in some epochs.

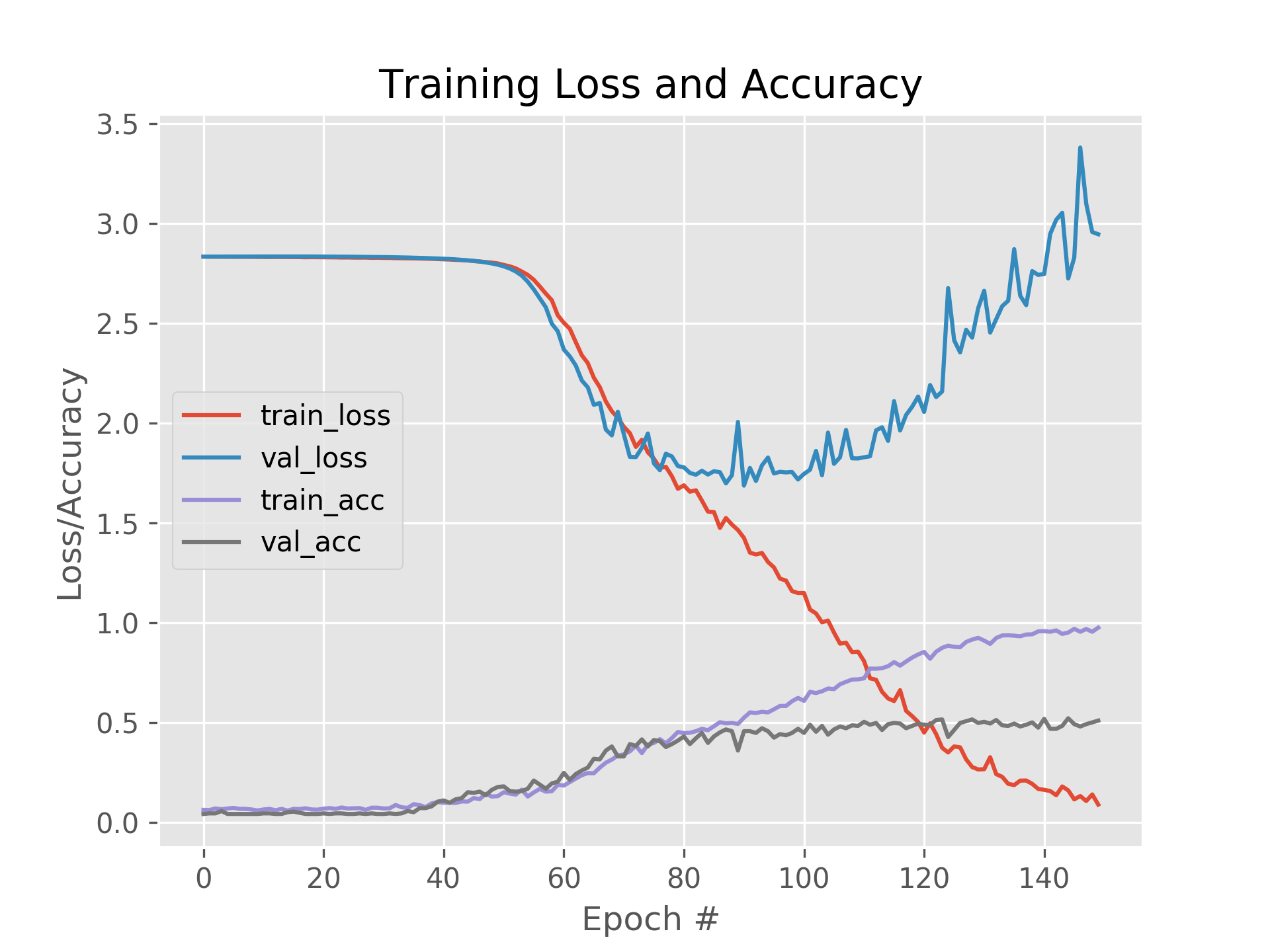


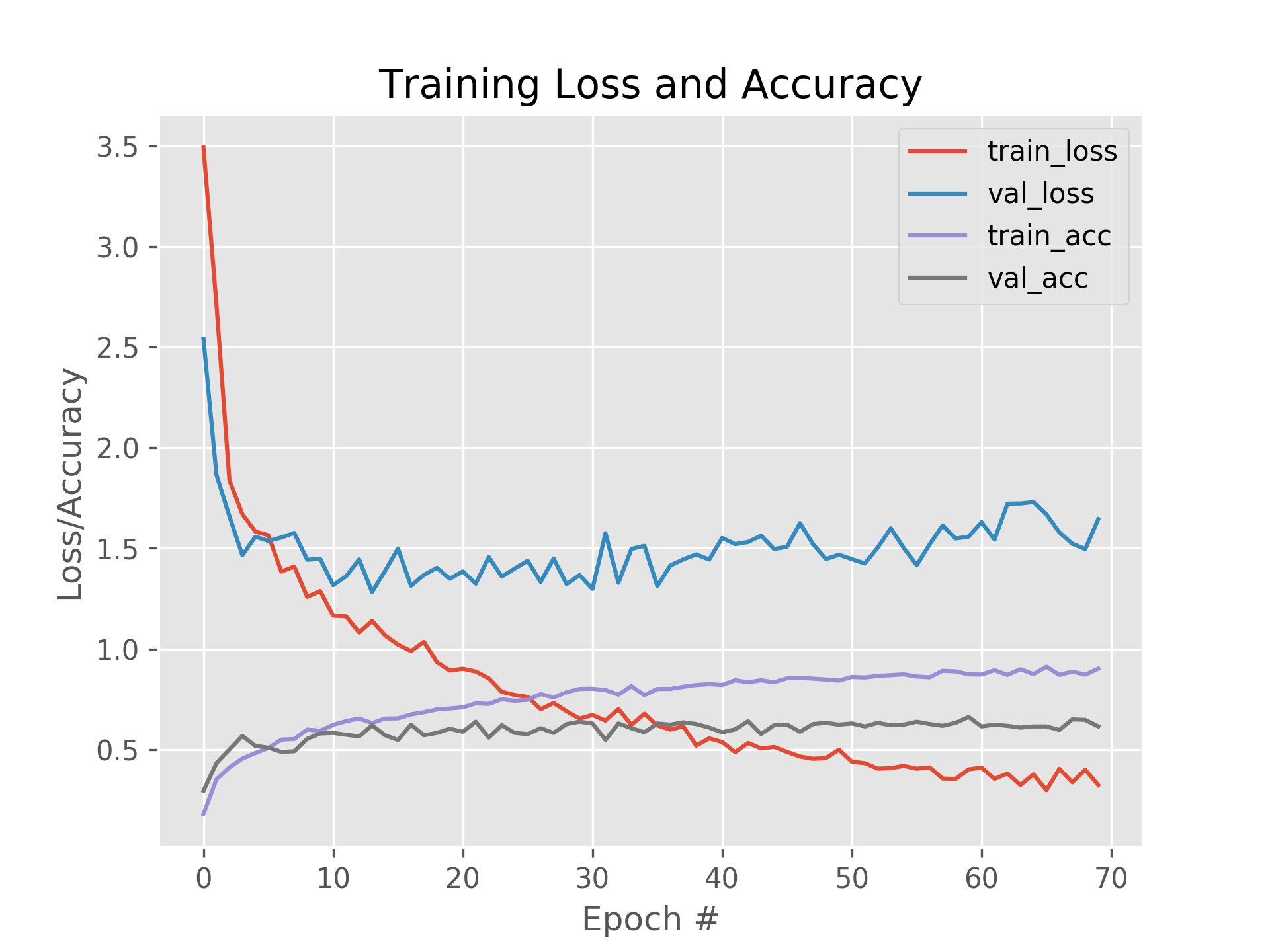
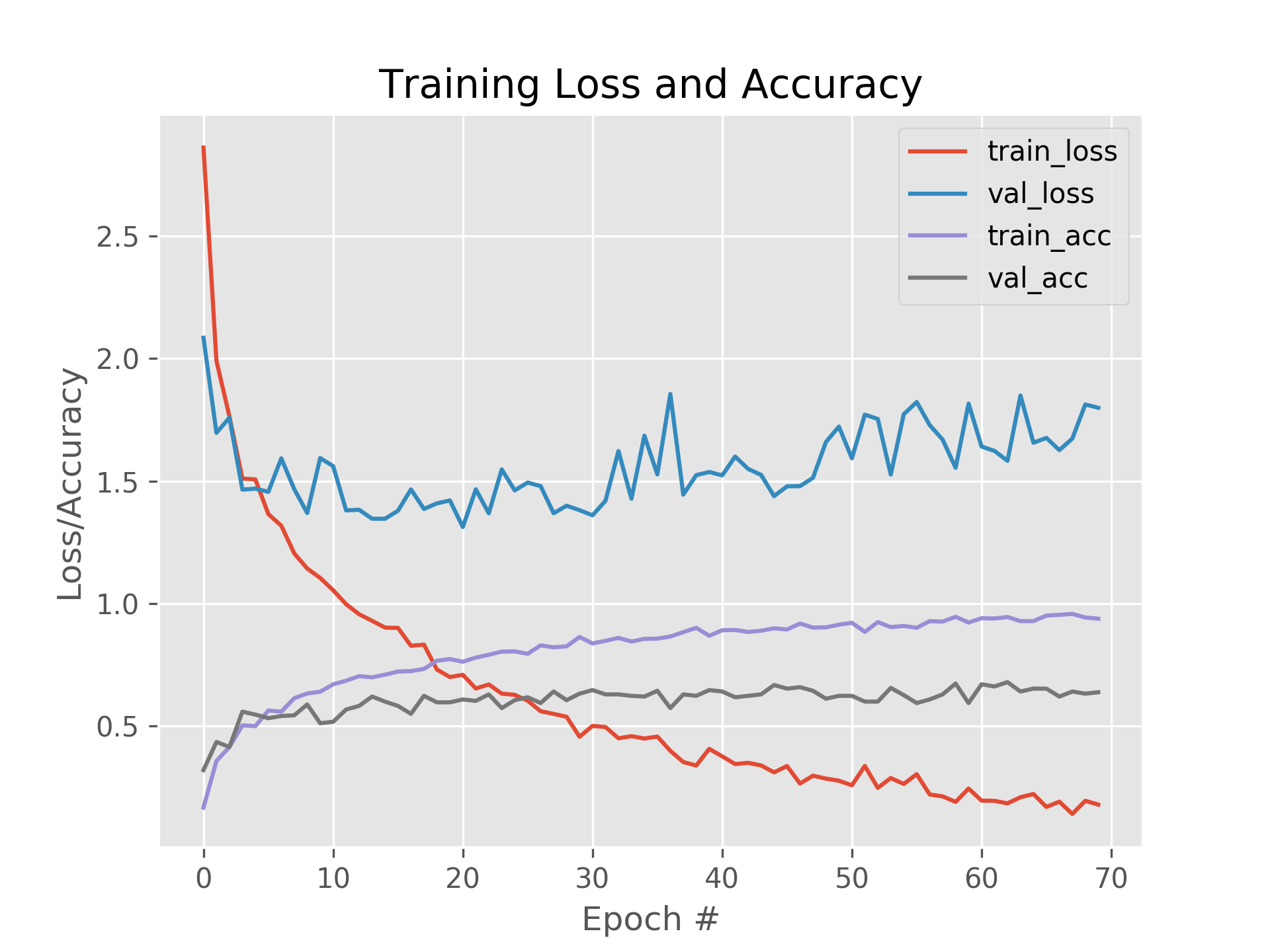
Fig 5: Config 3, 150 epochs, filter sizes 64, 128, 192 & 256, val accuracy 51%

* Further testing with different filter sizes for each layer over the 150 epochs for config 3, led us to the validation accuracy of 51% after 120th epoch, but, validation loss started showed the same pattern after the 60th epoch as it did for the previous configurations.
* All the configurations were thorougly tested for filter sizes of 64, 128, 256 and 512 for respective layers. Also with all layers having same number of filters and with dropout enabled and disabled.
* Dropout only showed insignificant improvement in valivation accuracies for configs 2 & 3, and bit smoother convergence curves.
* Higher filter sizes like 512 and beyond had negative impact on validation loss and accuracy curves, so 256 was chosen as the optimal filter size for the last layer of config 3.

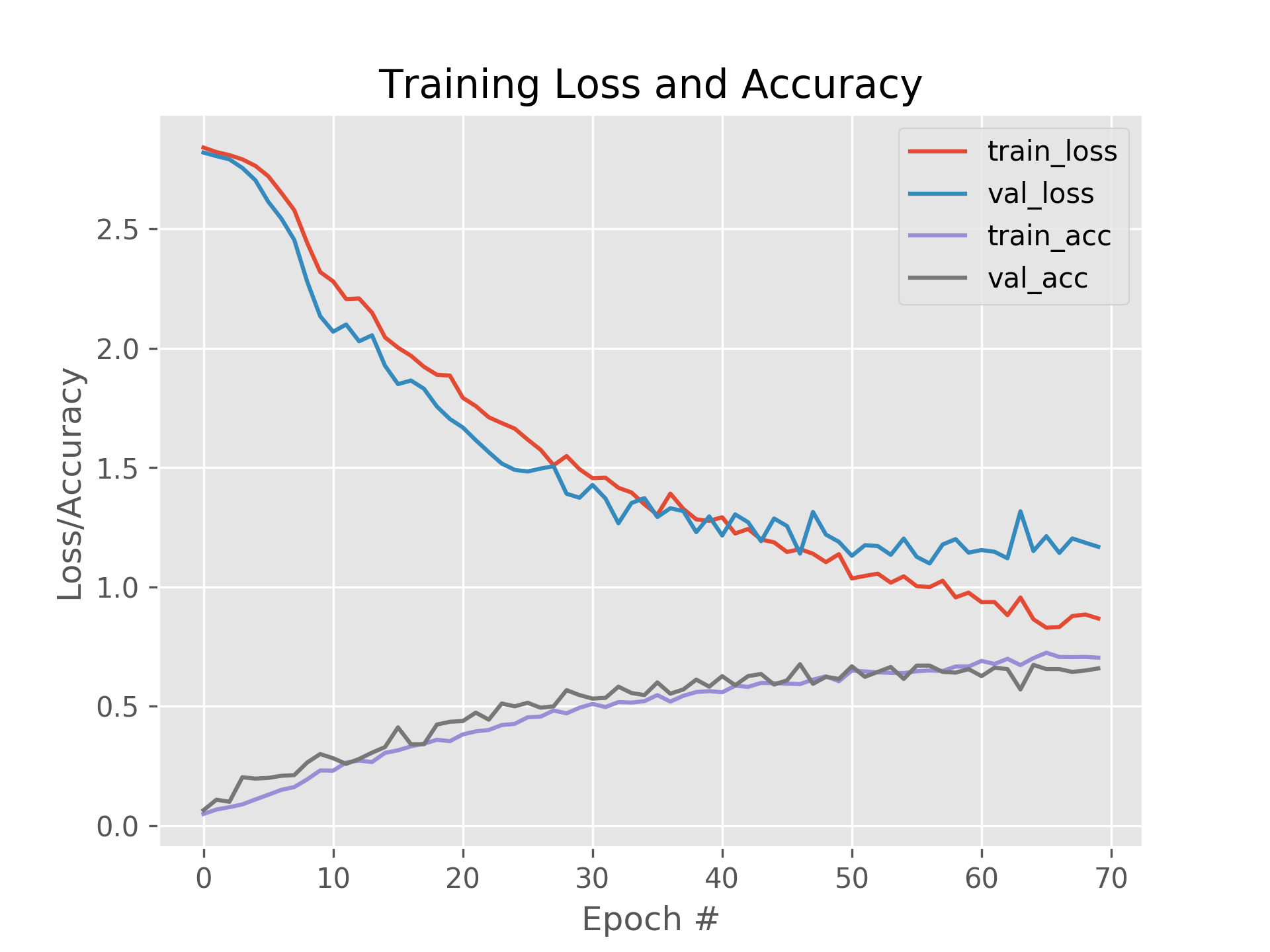
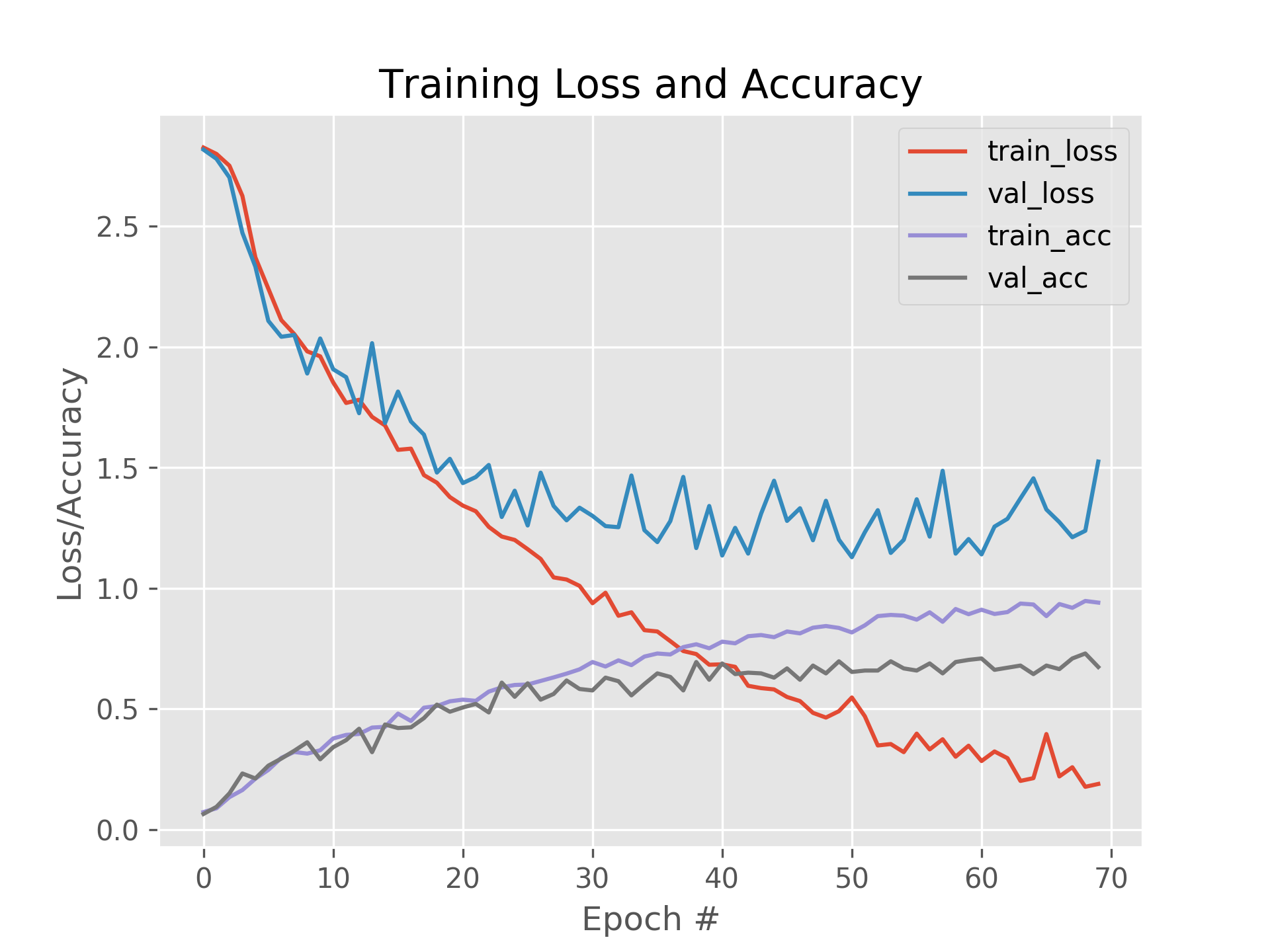
The existence of similar pattern in validation loss despite dropouts and different filter sizes, and similar pattern overfitting pattern for validation accuracy as well, with very slight improvement over 150 epochs strongly suggests that the training data has insufficient variety.

**Task (ii) Impact of Data Augmentation:**

* Data Augmentation generates new images by introducing variety such as vertical/horizontal flipping, rotations at specific degrees, zoom in/out, resize and normalization with features to fill missing pixels for resized images and few more.
* I tried all above mentioned varieties together and observed that, validation losses across baseline configs and config 1, 2 & 3 were significantly reduced down below 1.7 for the baseline model and upto 1.4 for config 3.
* The validation accuracies significantly improved for all models.
  + From 40% to 60% for the baseline model
  + From the range of 38 to 45 % to the range of 55-70% for config 1, 2 and 3 respectively.
  + Introducing the dropouts in each config models further regularized the validation loss curves and validation accuracy curves as shown below, especially better convergence for config 3.



Baseline, Augmentation, no dropout Baseline, Augmentation, with dropout



Config 3, Augmentation, no dropout Config 3, Augmentation, with dropout

* As it can be seen in above figures, augmentation improved the accuracy for all models and reduced the validation losses significantly.

**Influence on accuracy from the selection of methods such as cropping, flipping etc:**

**Only Vertical Flipping:**

* Only vertical flipping as augmentation choice, shown below, dropped the accuracy approx by 2% over the baseline, but, we see the drop in validation loss from 3 to 2.25, which is better than.
* The training accuracy is seen to continue to improve and doesn’t vary away too much from the validation accuracy, as it does without augmentation.

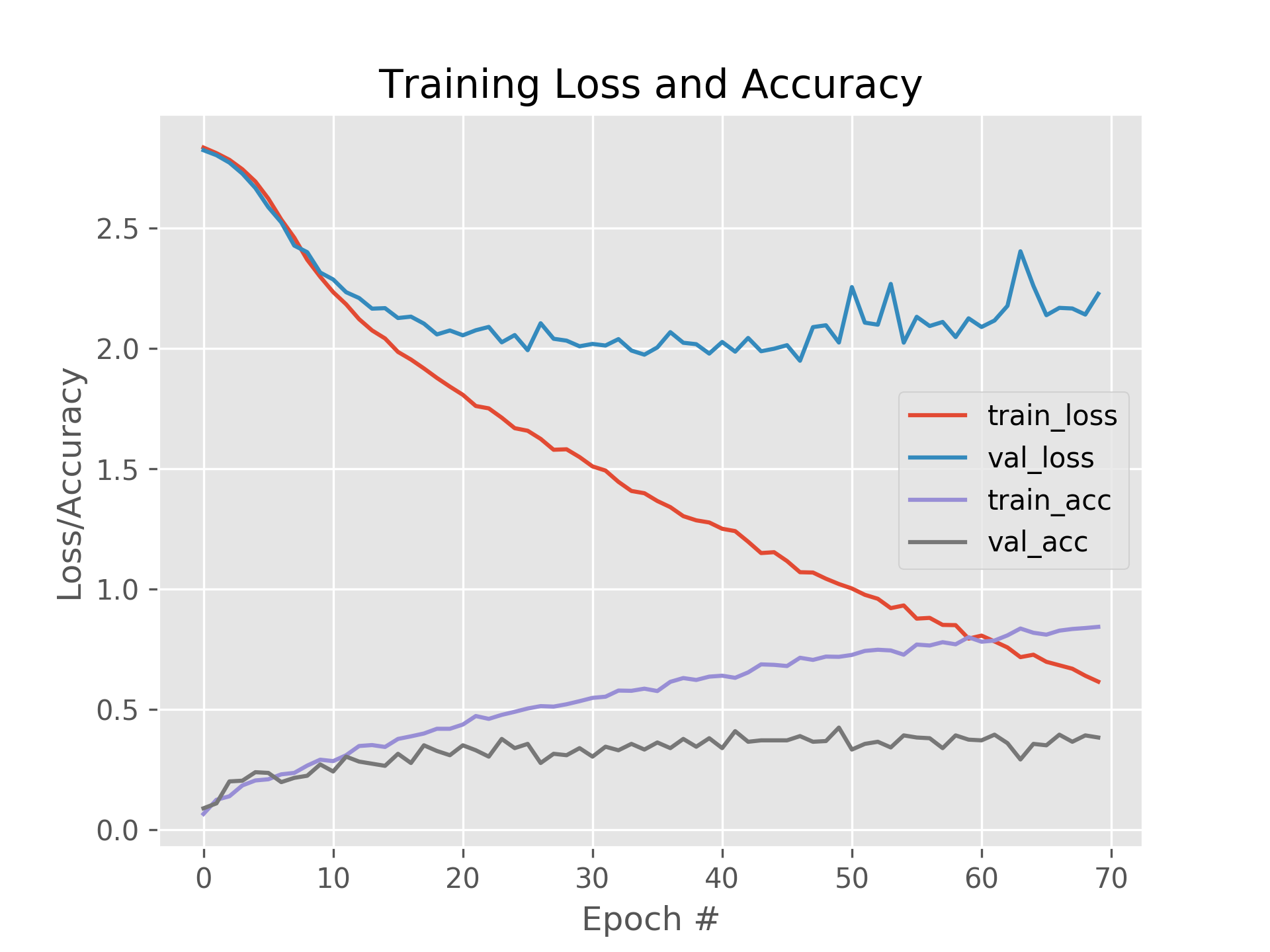


Fig: Vertical Flipping, Baseline model, Accuracy 38%

**Only Horizontal Flipping:**

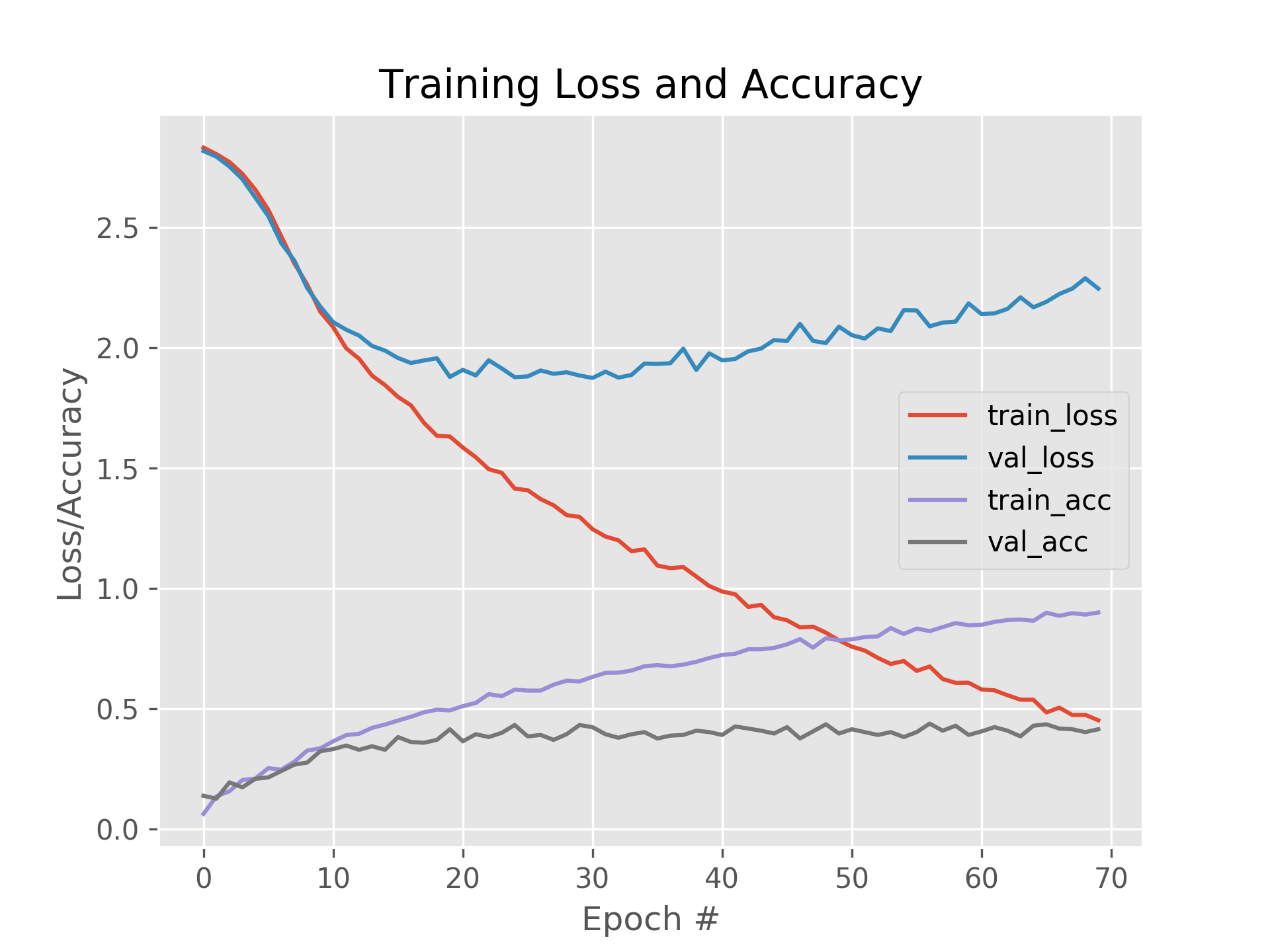


Fig: Only horizontal flipping, accuracy 41.7%

* Only horizontally flipping made validation loss curve much smoother over the vertical flipping and improved the validation accuracy to 41.7%, although combined vertical and horizontal flipping had accuracy around 40%. Hence, declaring horizontal flipping as best flipping suitable for the type of images that exist in the test set.

**Only Positive Zoom (20%):**

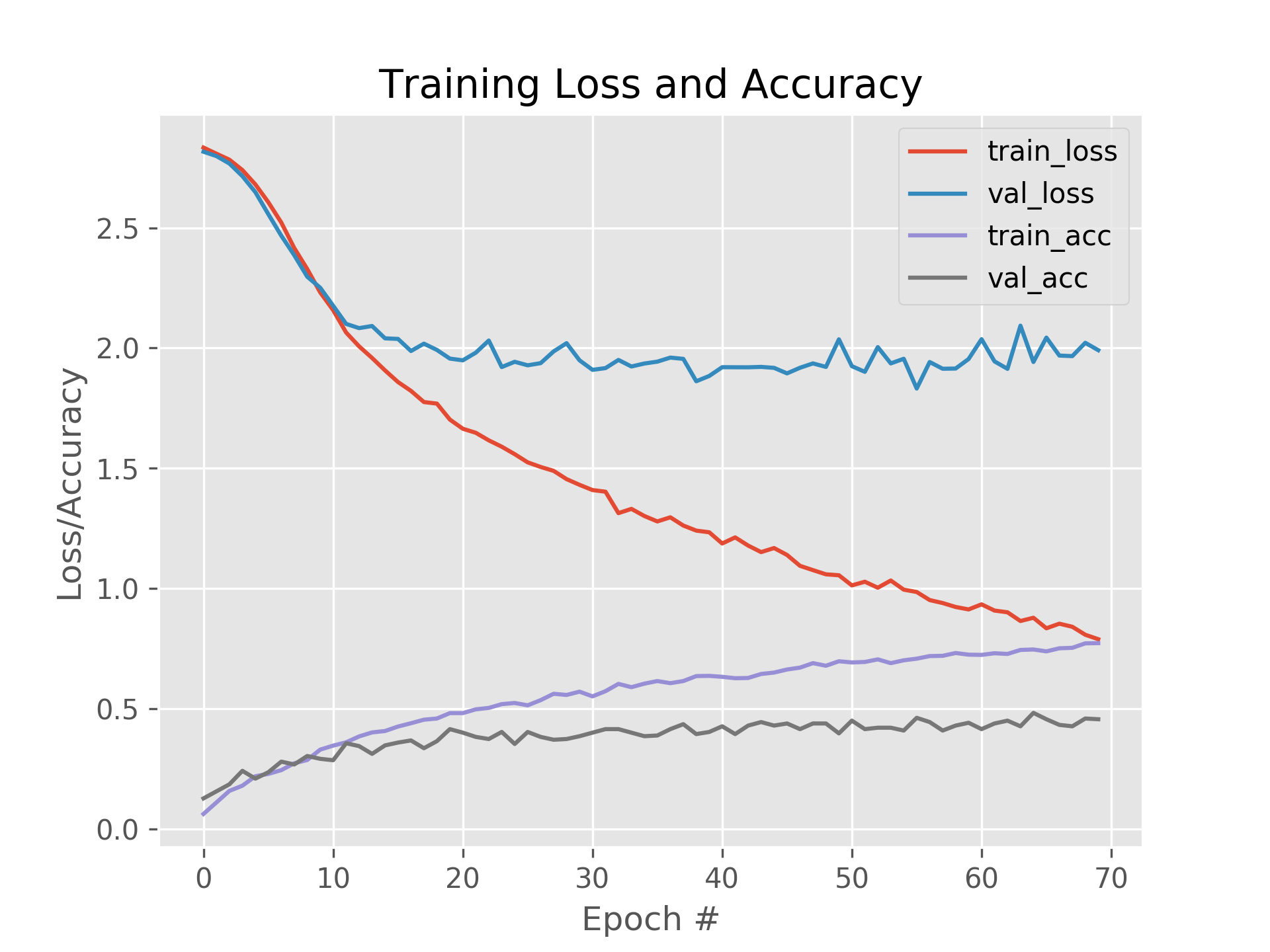


Fig: Only Positive zoom, accuracy 45%

* Choice of only Zooming in the image by 20% shown significant accuracy improvemet, 45%. Also the validation loss has dropped to 2.0, better than just flipping, as suited to this dataset.

**Only Negative Zoom (-20%):**

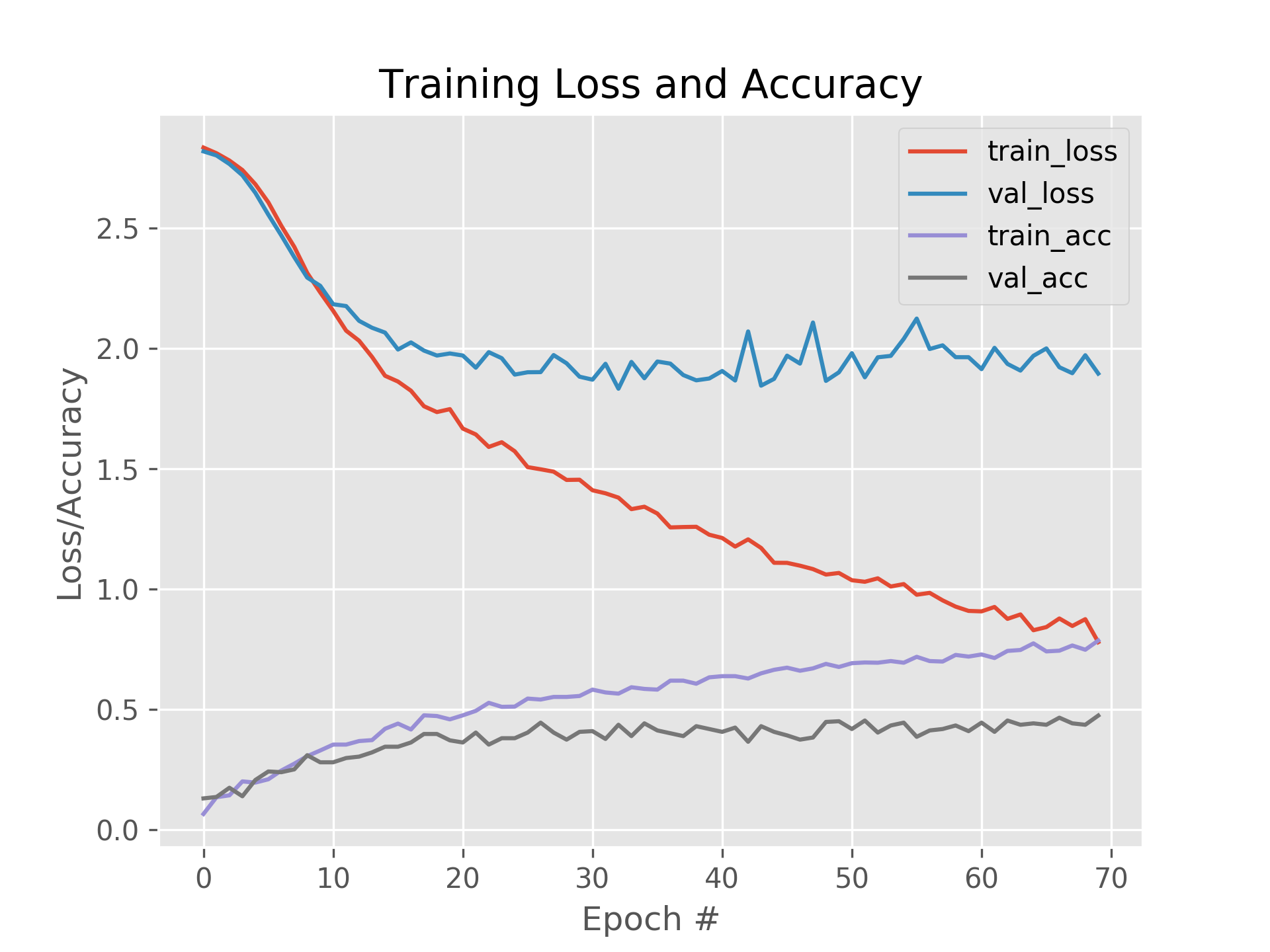


Fig: Only negative zoom, accuracy: 47.35%

* Only negatively zooming, i.e. zooming out the image resulted in improved accuracy of 47.35%, and drop in validation loss to 1.89
* Further combining negative zoom with 20% shear improved accuracy to 48.5%



Fig: Negative Zoom with shear and vertical flip, 150 epochs, accuracy: 51.76%

* As vertically flipping the image combined with shear and negative zoom resulted in reducing validation loss and improving validation accuracy, 150 epochs yielded 51.76% accuracy with some epochs 52.65% being the highest accuracy for intermediate epochs.

**Task (iii): Ensembles (8 base learners)**

Took 8 base learners by introducing some diversity in 4 models built in previous tasks of this assignment. The image below shows the validation accuracies for the individual models.

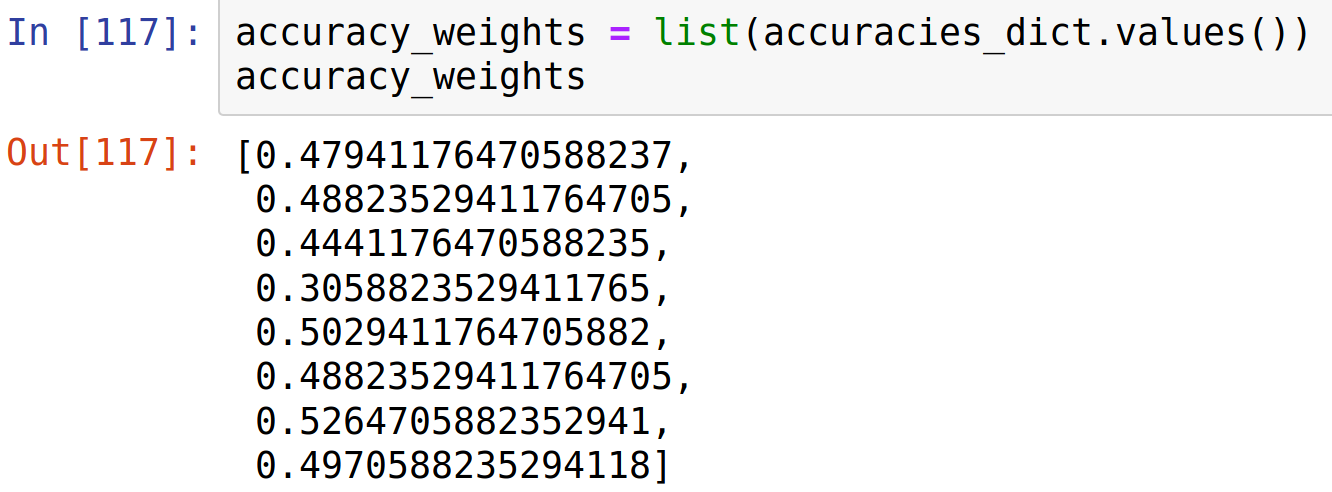
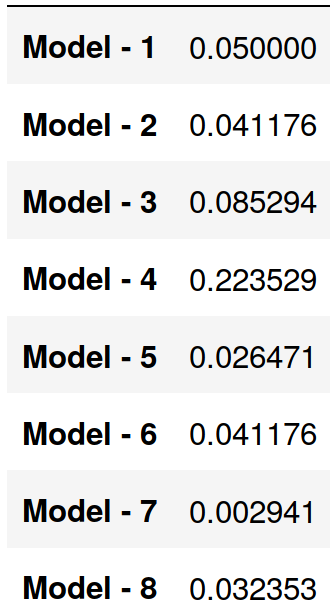


Fig: Validation Accuracies for 8 models

Results were aggregated using the Weighted Average techique which yielded 52.64%, where weight for each model is the validation accuracy as given in above image. The average without weights was slightly better, 52.94%

Due to the randomness in diversity, as explained in answers to related questions below, some runs also resulted approx 56% accuracy.



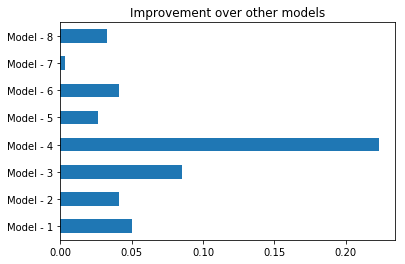


Fig: Ensemble Improvement over each model

Above images show the comparison about how much improvement rate did Ensemble average method offer over the accuracies of individual models.

**Why might lack of diversity be a problem of ensemble techniques?**

Given multiple models of similar kind to ensembles, some models might be better at recognizing the images of only certain classes, or better at recognizing only certain aspects of some classes of image while the remaining aspects can be addressed well with different models of different architecture in terms of regularization, image augmentation and more. But, lets say, if ensemble consider models only good at recognizing german shepherd and husky dogs, where actual test images contain equally high volume of other diverse types of dogs, the overall accuracy would suffer rather than improve.

**Can you describe possible steps you could take to introduce additional diversity into the base models?**

For this assignment, I only took random values as shown in below image for data augmentation technique, varied number of filters for each layers for different models, and enabling/disabling regularization. They may not be very sensible combinations or not necessarily perform well. But, below steps could introduce enough sensible diversity in the base models.

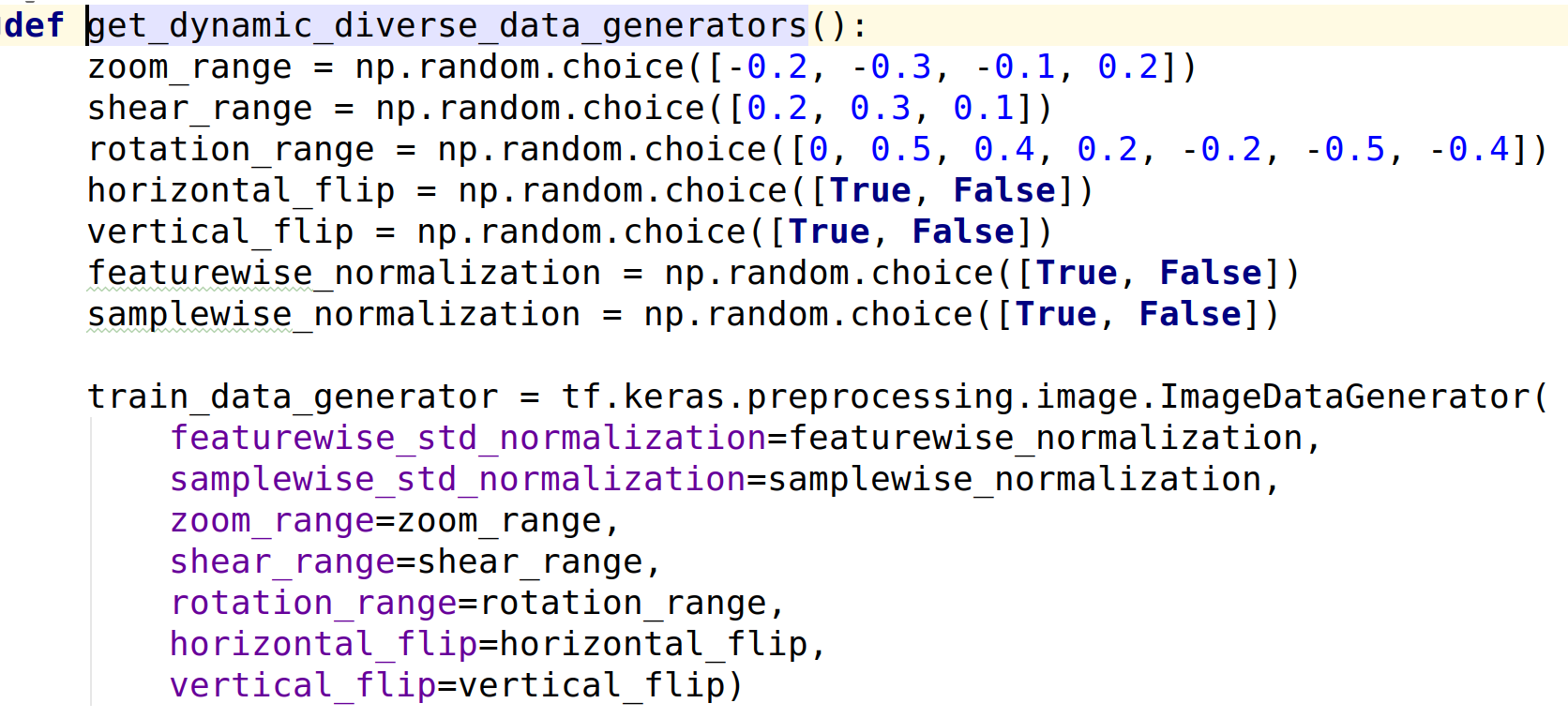


Fig: Diversity in Data Augmentation using random values

1. Understand the effect of different values of batch sizes, filter counts and sizes, nature and number of convolutional/dense layers, data augmentation parameters like zoom\_range, flips, cropping, the ones shown in above image and other possible augmentation parameters.
2. Using different possible values across all models, predict results and look at the precision, recall and f1 scores for individual model predictions to know what configurations work best for which features and classes.
3. From the notes of step 2, choose such a different model configurations that individually offer good classification for certain classes, making sure to choose such a different models, that, when combined, overall ensemble should yield better final f1, precision and recall scores across classes.

**Part B: Transfer Learning**

**Task i)**

I tested different base models VGG16, VGG19 and Inception3 with LogisticRegression and RandomForestClassifier.

For all 3 base models, RandomForestClassifier yielded best accuracy of 65% with these parameters:

n\_estimators=600, max\_depth=12

For all 3 base models, LogisticRegression yielded the best accuracy of 72.05% to 73% using the ‘lbfgs’ and ‘sag’ solvers respectively.

Therefore, LogisticRegression was the best prediction model in comparison to the RandomForestClassifier.

**Task ii)**

Configuration for following tests:

Learning Rate: 0.001

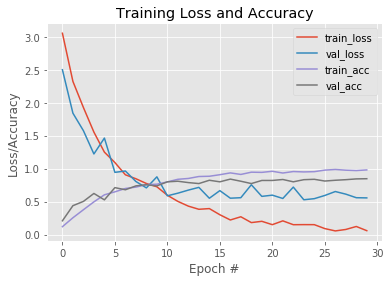
epochs = 30

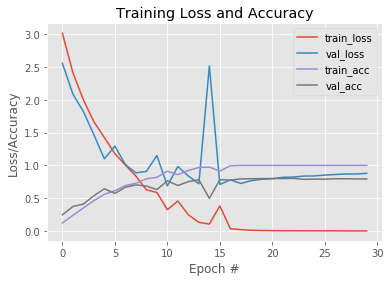
**Model1:** Base Model: Vgg16 -> 500 Dense relu -> (optional configurable) Dropout 0.2 -> 250 Dense relu -> (optional configurable) dropout 0.1 ->

17 softmax units

**Only Dense Trainable Layers:**

**Without Dropout With Dropout**

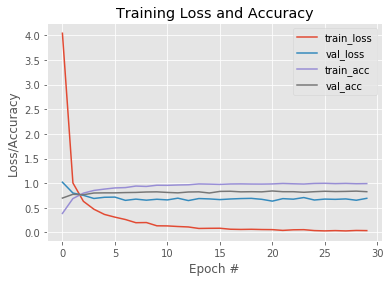
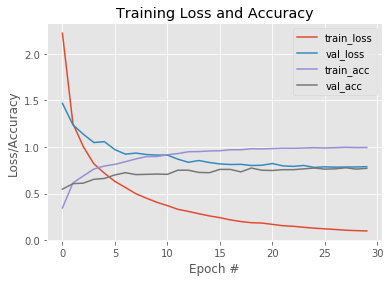
****

****

Validation Accuracy: 79.41% Validation Accuracy: 84.71%

**‘block5\_conv3’ onwards trainable:**

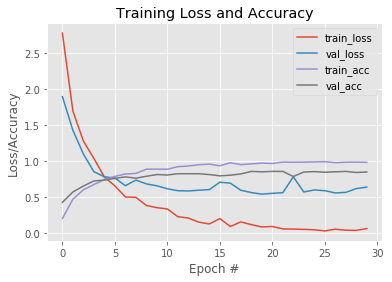
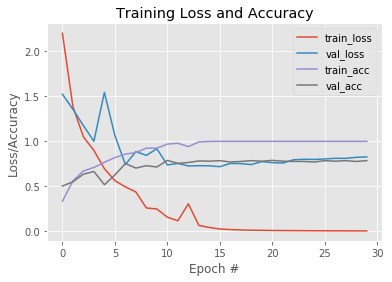
**Without Dropout With Dropout**



Validation Accuracy: 77.06% Validation Accuracy: 82.65%

**‘block4\_conv1’ onwards trainable:**

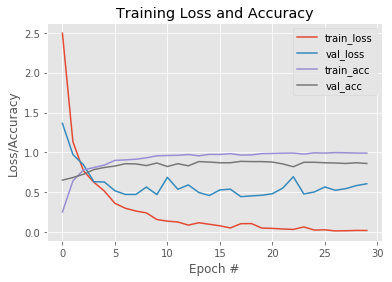
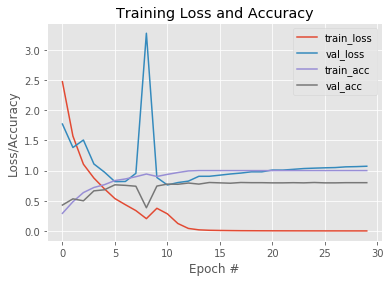
**Without Dropout With Dropout**

****

Validation Accuracy: 78.53% Validation Accuracy: 85%

**‘block3\_conv1’ onwards trainable:**

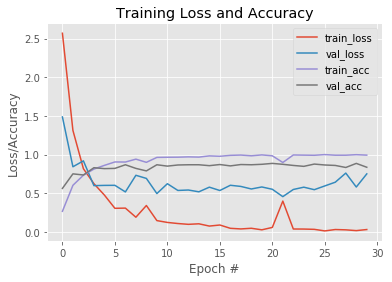
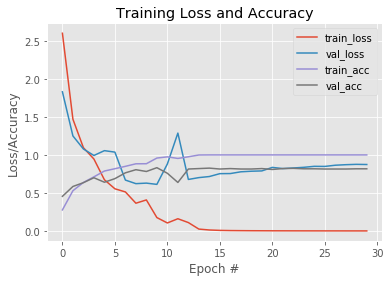
**Without Dropout With Dropout**

****

Validation Accuracy: 80% Validation Accuracy: 86.18%

**‘block2\_conv1’ onwards trainable:**

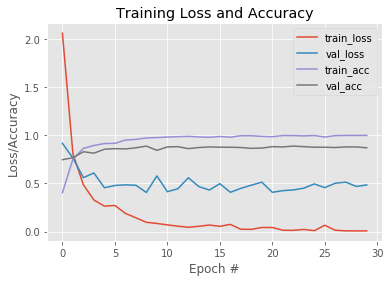
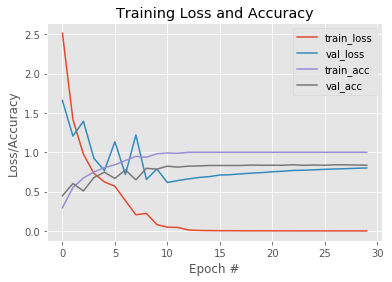
**Withrout Dropout With Dropout**

****

Validation Accuracy: 81.76% Validation Accuracy: 83.53%

**‘block1\_conv1’ onwards trainable:**

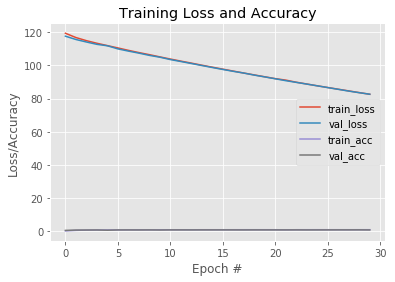
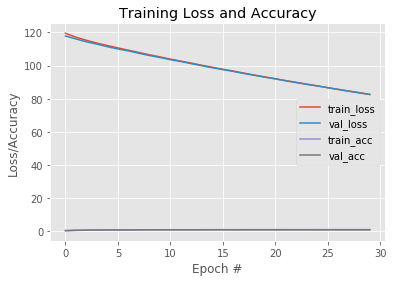
**Without Dropout** **With Dropout**



Validation Accuracy: 83.53% Validation Accuracy: 87.06%

**With L2 kernel/weights regularization (0.1% and 0.07% resp. for 2 Dense layers):**

**Freezing all VGG16 layers Unfreezing from block1\_conv1**

****

Validation Accuracy: 85% Validation Accuracy: 85.29%

Further experiments with different number of Dense layers and regularization yielded accuracies around 91%.

**Part C: Research**

**Capsule Networks**

CNN’s are good at learning good features in certain areas of the image, but, they aren’t capable of relating tiny features hierarchically and linking the likely features together to predict the bigger objects.

**Capsule Networks** are the hierarchical networks of other capsules, just like in a tree structure a.k.a. Parse trees[1], where lower level capsules represent the smaller parts of the entire object/entity. The lower level capsules consider the orientation of the object to predict which higher level objects it belongs, the length of the object, and certain other things such as *pose*, deformation, velocity, texture, and more to predict if the existence and association of the entity to the object in the parent capsule. When all lower level capsules agree, the parent capsule becomes active, which then considers similar parameters for object higher in the hierarchy, and activating their respective parents upon the agreement and so on. This is how, capsule networks help us recognize the objects in images despite certain distortions. As claimed by Sara et. al. in [1], ‘it achieves state-of-the-art performance on MNIST dataset.’

Output of low level capsule, the vector, when routed to their respective parents, is scaled down by a coupling coefficient. The coupling coefficient of higher level capsules is their magnitude, the larger the coefficient, higher the chances for the object belonging to the category being represented by that particular capsule tree/hierarchy. The parents at the same levels across all the hierarchies/trees multiply their output with the weight matrix to compute the scalar product. Whichever parent has a highest value for the scalar product, increases their own coupling coefficient using the top-down feedback approach; at the same time, lowering the coupling coeffient for other top level parents [1]. In short, this process increases the magnitude of the capsule that has the closest resemblance to the expected object, increasing the chances of actual object being identified correctly.

**Acknowledgements:**

1. Thanks to Dr. Ted Scully for his intuitive reference material, and enthusiastically teaching this subject and creating so much interest about it. Parts of the code snippets for this assignment are referred from his lecture slides.
2. Thanks for the best open frameworks, Scikit Learn [2], Tensorflow [3], Keras [4]. Some code snippets were also referred from these sources.

**References:**

[1]Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic routing between capsules." *Advances in neural information processing systems*. 2017.

[2] Buitinck, Lars, et al. "API design for machine learning software: experiences from the scikit-learn project." *arXiv preprint arXiv:1309.0238* (2013)

[3] TAbadi, Martın, et al. "TensorFlow: Large-scale machine learning on heterogeneous systems, 2015." *Software available from tensorflow. Org* 1.2 (2015).

[4] Chollet, François. "Keras." (2015).