Itamar Epstein 203253145

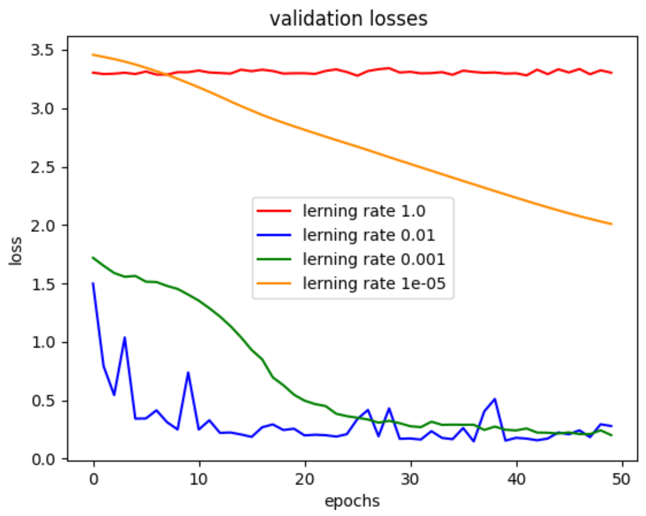
Machine Learning Methods

Exercise 4

6 Multi-Layer Perceptrons

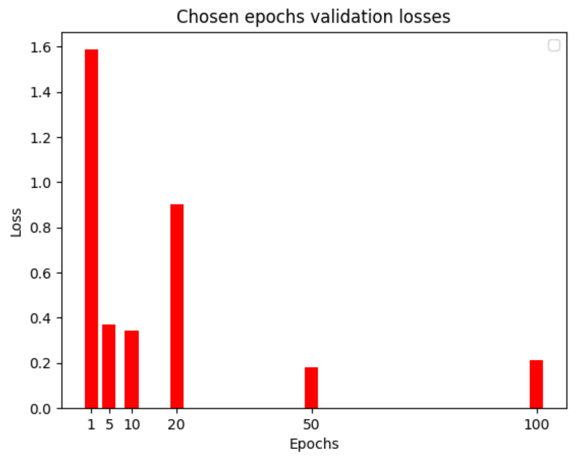
6.1.2 Questions

1. What does a too high learning rate do? What about a too low one? Explain.



The learning rate is a crucial hyperparameter in training neural networks. It determines the size of the steps taken during the optimization process.
A high learning rate, can lead to several detrimental effects. Firstly, it carries the risk of overshooting the optimal solution, and also to bypassing the lowest point in the valley entirely. This translates to the network failing to converge on the minimum as exemplified in the plot by the red line with a learning rate of 1, the model encounters difficulties in learning and locating the minimum of the loss function. The optimizer's steps become overly large, preventing the model from reaching optimal values. There for the line stays sright and does not improve. (can also explain the fluctuation of accuracy values on the test and validation sets)
in the other hand a learning rate set too low, such as 0.00001, presents a different challenge. In this scenario, the steps are excessively small, resulting in slow progress in minimizing the loss function. Although positive progress is made, achieving satisfactory accuracy within 50 epochs becomes impractical. What can be seen in the plot, the orange line is alwyes improving, but is far from a good result

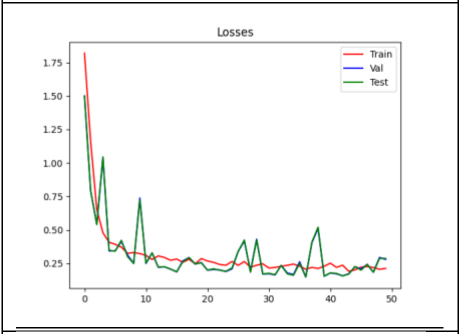
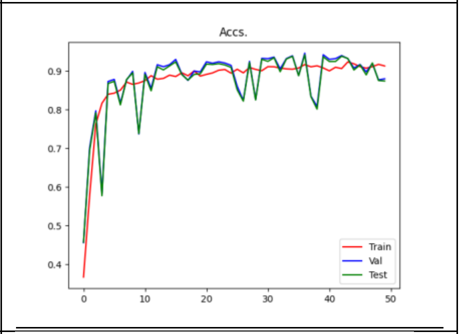
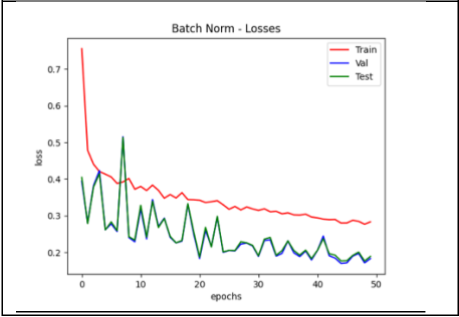
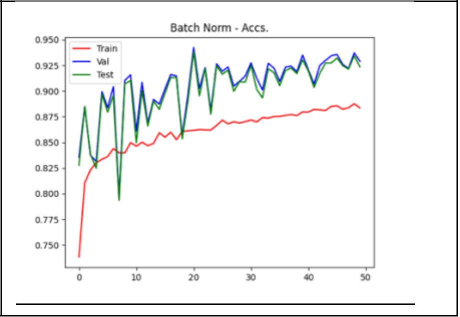
2*. How does the number of epochs affect the training? What happens after too much epochs? How does too little epochs affect? Explain.*



The number of epochs significantly influences the training dynamics of a neural network.
It's essential to consider that, generally, training the model for more epochs tends to result in a **smaller loss** value for the training set, contributing to better accuracy.
However, an excessively well-fitted model to the training set may lead to overfitting, adversely affecting generalization to new data and resulting in worse test and validation accuracies.
Like can be seen in the plot, in the early epochs (1-10), there's a positive correlation between the increase in epoch count and improvements in training loss (as well validation, and test accuracies), indicating effective learning and generalization.
However, during the mid epochs (20-50), a subtle dip in the loss score at epoch 20 suggests potential early signs of overfitting or a shift in data distribution. Despite this, the test accuracy remains relatively stable. As we progress to later epochs (100), we observe a decrease in the loss value of the validation set until epoch 50, after which the loss value increases at epoch 100. This dynamic indicates a potential overfitting of the training data.
In summary, the amount of epochs with the loss function is pivotal: too few epochs may result in poor accuracies due to insufficient minimization of the loss function, while too many epochs may induce overfitting and affect the generalization. it's really important to find the right number of times to train the model, (epochs), to get the best performance. This is especially crucial when looking at how well the model reduces its mistakes, or loss, during training.

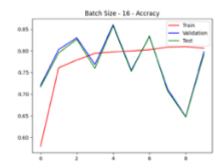
3. Compare the results of the regular and modified model as before. How does the batch normalization layer contribute the training? Explain.

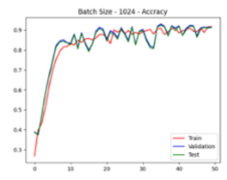
Adding a batch normalization layer (nn.BatchNorm1d) after each hidden layer in the neural network can have a significant impact on training. Batch normalization contributes to the training process in several ways, firsty When we look at the graphs, it's can be seen that the model with batch normalization achieves higher accuracies for both the val and test sets. However, it has a lower accuracy for the training set alone. This suggests that batch normalization helps the model to generalize better from the training set, improving its performance on unseed data. The idea is Normalization layers assist in evaluating features not only based on their high values but in comparison to their typical values. This technique is likely to enable the model to learn more generalizethion from the training set. Secondly, When we check losses, we can see that the model with normalization improves a bit less. However, It can prevent the model from learning too quickly or becoming too focused on specific details, which could be erelevant or not very important. So, while normalization may not always make the model improve faster, it contributes to a more generalize model.

LOSSES	ACC
	
	

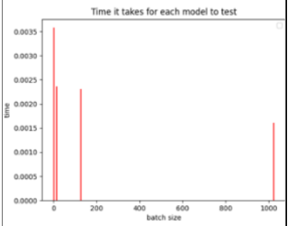
4. How did this choice affect the (i) accuracy? (test) (ii) speed? (test) (iii) stability?).

(i) Accuracy - The impact of varying batch sizes on test accuracy is notable. With a batch size of 1, the model's accuracy might suffer due to erratic updates and challenges in generalization. As the batch size increases there's a potential improvement in accuracy as the model benefits from more stable updates. Therefor when we use larger batches for training, we tend to get more accurate results. This is because larger batches make it less likely to have erelevant or not very important data that could affect how the model learns. So, when models are trained with bigger batches, they are expected to perform better on the test set, as these batches represent the data more reliably.

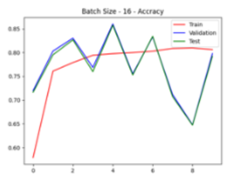


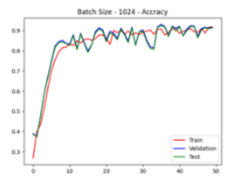


(ii) Speed - The choice of batch size significantly influences the training speed. A batch size of 1 results in a slow training process due to frequent model updates for each individual data point. Increasing the batch size accelerates training by processing more data in each update, finding a balance between speed and stability. The consistent evaluation speed on the test set across different models contrasts with the increase in learning speed as batch sizes grow. This discrepancy can be explained by the fact that larger batch sizes lead to more efficient learning. The model processes more data in each update, necessitating less computing for weight adjustments each epoch. As a result, while learning speed increases.



(iii) Stability - The stability of the training process, reflected in the noisiness of the loss curve, varies with batch size. A batch size of 1 leads to a highly noisy training process, with frequent fluctuations in the loss, potentially hindering stability and generalization. Increasing the batch size generally results in a more stable loss curve, allowing for smoother convergence and reduced noise. The observed decrease in "noisiness" of the loss graph with increasing batch size aligns with the notion that larger batches contribute to a more balanced dataset. As the batch size grows, the probability of encountering a biased batch diminishes. This results in more balanced data, which in turn leads to a smoother loss curve over epochs. The reduced noise in the loss graph indicates a more stable training process, supporting better fitting to the test and validation data.





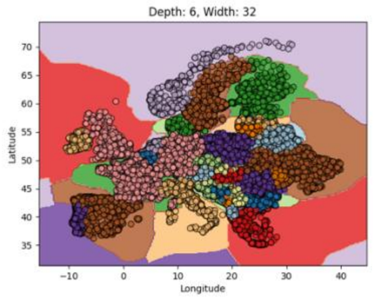
6.2 Evaluating MLPs Performance

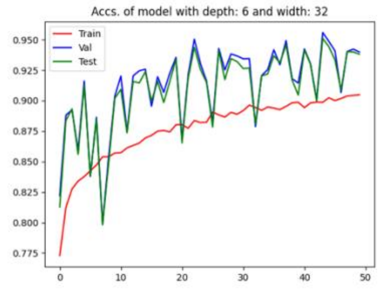
6.1.2 Questions

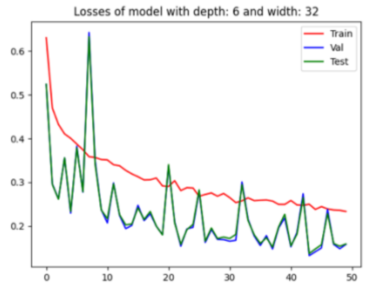
1. Choose the model with the best validation accuracy , Did this model generalize well? Explain.

The model with the highest validation accuracy is: (0.9398338696400509, 6, 32)

Chosing the model with the highest validation accuracy, (0.9398338696400509, 6, 32), did well in the generalization performance. A high validation accuracy indicates that the model performed well on the validation set, showcasing its ability to make accurate predictions on data it hasn't seen during training. In Adithion a low loss factor on the val/test sets what can be seen in the - loss plot shows that the model did well on the new data,in conclusion In the graph you can see that the loss values of the test and validation sets are smaller than the training loss value and that means that the model generalized well from the training set.



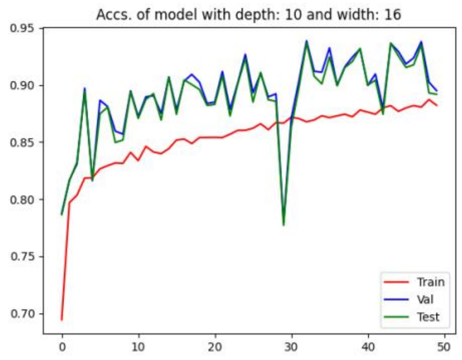


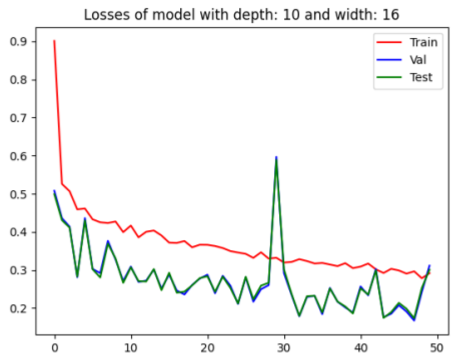


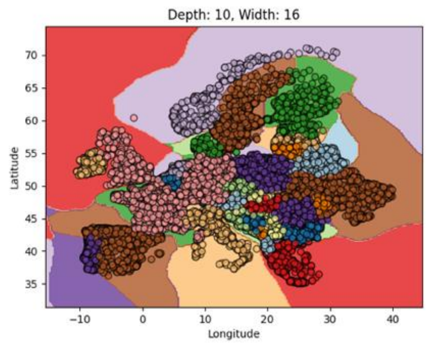
2. Did this model over-fit or under-fit the dataset? Explain.

The model with the lowest validation accuracy is: (0.8949337723565067, 10, 16)

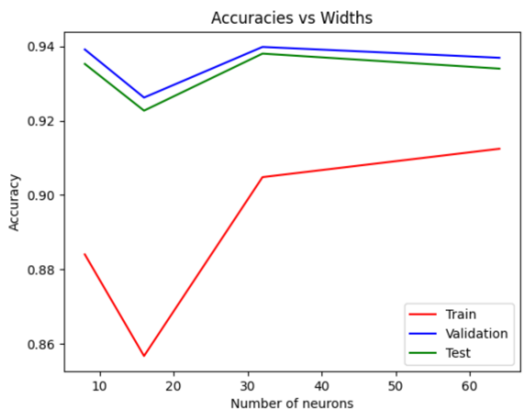
The he worst model we encountered ended up fitting too closely to the training data, as evident in our plots. Even though it performed well on the training set with an accuracy of almost 0.9, it struggled on new data, resulting in poor performance on the test set. Additionally, this model showed better results on the training loss compared to the validation and test sets, indicating signs of overfitting. Notably, when we compare its performance with the best model having a size of 6 layers and 32 units, the worst model with 10 layers and 16 units exhibited a significant gap in generalization. The difference in size, specifically the increase in the number of layers and decrease in the number of units, can impact generalization. A larger model with more layers and fewer units per layer has the potential to capture intricate patterns and details in the training data, leading to overfitting. This overfitting occurs when the model becomes too specialized in learning the nuances of the training set, hindering its ability to generalize well to new or unseen data. In contrast, the more balanced architecture of the best model with 6 layers and 32 units strikes a better equilibrium between capturing essential patterns and avoiding overfitting, contributing to its superior generalization performance.





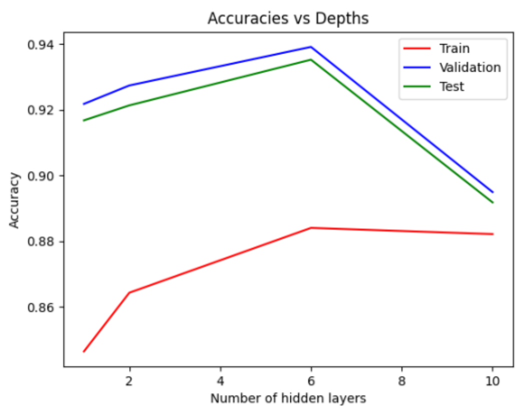


<u>3 . How does the number of layers affect the MLP? What are the advantages and disadvantages of increasing the number of layers in a neural network? Explain.</u>



Adding more layers to a network helps it understand complex data patterns, potentially improving performance. Each layer acts like a tool, transforming data in a way that builds intricate representations. However, this advantage can be tricky, especially with limited training data. Too many layers might make the network memorize specific details from training instead of learning general patterns, causing overfitting. In our experiment, the graph shows that having either too many or too few layers doesn't help model accuracy. While more layers allow for more data processing and improve training accuracy, it also risks over-processing and overfitting to the training data, leading to worse validation and test accuracy. Our simple experiment highlighted that adding layers too quickly resulted in worse accuracies, underlining the importance of a balanced approach in designing the model.

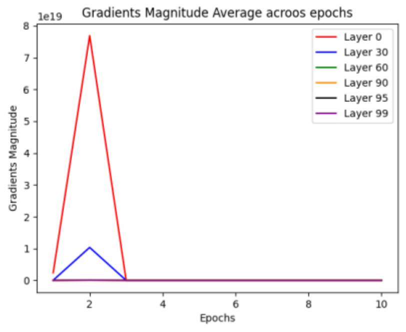
<u>4. How does the number of neurons affect the MLP? Explain.</u>



The number of neurons in an MLP influences its performance. Having more neurons allows the network to capture intricate details and patterns in the data, potentially improving overall performance. Each neuron acts like a small processing unit, contributing to the model's ability to understand complex relationships. However, too many neurons can lead to overfitting, especially if the dataset is limited. Overfitting occurs when the network becomes too focused on memorizing specific details from the training data, what is more often with a lot processing units , hindering its ability to generalize to new or unseen data. Our experiment highlighted that simply increasing the number of neurons did not result in positive or significant changes in the validation and test set accuracy(the train acc improved – what can show of memorizing specific details from the training data) .

This underscores the importance of thoughtful consideration when determining the number of neurons to strike a balance between capturing essential patterns and avoiding overfitting.

<u>5. Do you see any problem of vanishing or exploding gradients? If so, what could we modify in the network, other than its depth, to solve this problem? Explain.</u>



Exploding gradients suggest that we might be far from reaching the minimum of the loss function. However, vanishing gradients may indicate that we are close to the minimum.

When far away, we want to take bigger steps in our learning process, while being close requires smaller steps. A solution to this issue involves using the gradient clipping technique. By setting a threshold on gradient values during training, we ensure that if a gradient surpasses this threshold, it is scaled down. This prevents gradients from becoming excessively large and helps maintain stable and effective learning in the neural network. in our case, inspecting the "Gradients Magnitude Average Across Epochs" plot reveals potential vanishing gradient issues in layers 60, 90, 95, and 99, where the average magnitudes persistently hover close to zero. Additionally, sudden jumps in the average gradient magnitude for layers 0 and 30 indicate potential occurrences of exploding gradients. To deal wuth these challenges without changing the network's depth, we can implement strategies like proper weight initialization, selecting activation functions resistant to vanishing gradients (e.g., ReLU), applying batch normalization, introducing gradient clipping, and dynamically adjusting the learning rate. These modifications aim to improve stability and encourage better gradient flow, addressing concerns associated with vanishing or exploding gradients in our neural network.

7 Convolutional Neural Networks

7.2 Task

1 . we'll experiment with the Adam optimizer and with different learning rates in the range of [0.00001, 0.1]

Results -

Scratch

Learning rate: 0.000010, Loss: 0.699190, Train accuracy: 0.506429, Val accuracy: 0.505000, Test accuracy: 0.520000

Learning rate: 0.000100, Loss: 0.702355, Train accuracy: 0.563571, Val accuracy: 0.475000, Test accuracy: 0.535000

Learning rate: 0.001000, Loss: 0.750572, Train accuracy: 0.543571, Val accuracy: 0.520000, Test accuracy: 0.505000

Learning rate: 0.010000, Loss: 0.942134, Train accuracy: 0.497857, Val accuracy: 0.455000, Test accuracy: 0.477500

Learning rate: 0.100000, Loss: 1.414474, Train accuracy: 0.532857, Val accuracy: 0.585000, Test accuracy: 0.515000

Liner

Learning rate: 0.000010, Loss: 0.703207, Train accuracy: 0.489286, Val accuracy: 0.500000, Test accuracy: 0.500000

Learning rate: 0.000100, Loss: 0.702342, Train accuracy: 0.566429, Val accuracy: 0.545000, Test accuracy: 0.582500

Learning rate: 0.001000, Loss: 0.646928, Train accuracy: 0.672143, Val accuracy: 0.645000, Test accuracy: 0.662500

Learning rate: 0.010000, Loss: 0.660453, Train accuracy: 0.766429, Val accuracy: 0.730000, Test accuracy: 0.715000

Learning rate: 0.100000, Loss: 2.643469, Train accuracy: 0.760714, Val accuracy: 0.745000, Test accuracy: 0.712500

Fine tuning

Learning rate: 0.000010, Loss: 0.622222, Train accuracy: 0.793571, Val accuracy: 0.690000, Test accuracy: 0.717500

Learning rate: 0.000100, Loss: 0.475448, Train accuracy: 0.970000, Val accuracy: 0.840000, Test accuracy: 0.837500

Learning rate: 0.001000, Loss: 0.632581, Train accuracy: 0.717143, Val accuracy: 0.675000, Test accuracy: 0.685000

Learning rate: 0.010000, Loss: 0.953656, Train accuracy: 0.500714, Val accuracy: 0.505000, Test accuracy: 0.510000

Learning rate: 0.100000, Loss: 1.948057, Train accuracy: 0.495714, Val accuracy: 0.510000, Test accuracy: 0.492500

1: What are your two best models from each baseline and your worst one in terms of test accuracy?
What learning rates did you use for the PyTorch CNN baselines? Explain why these are the trends you see.

Two best models and worst one from Scratch baseline:
Best Model: Learning rate: 0.000100, Test accuracy: 0.535000
Second Best Model: Learning rate: 0.000010, Test accuracy: 0. 520000
Worst Model: Learning rate: 0.010000, Test accuracy: 0.477500

Two best models and worst one from Linear baseline:
Best Model: Learning rate: 0. 010000, Test accuracy: 0. 715000
Second Best Model: Learning rate: 0.100000, Test accuracy: 0.712500
Worst Model: Learning rate: 0.000010, Test accuracy: 0.500000

Two best and worst one models from fine-tuning baseline:
Best Model: Learning rate: 0.000100, Test accuracy: 0.837500
Second Best Model: Learning rate: 0.000100, Test accuracy: 0.717500
Worst Model: Learning rate: 0.100000, Test accuracy: 0. 492500

PyTorch CNN baselines learning rates [0.00001,0. 0001 ,0.001, 0.01, 1]

From the observed "two best and worst one models" results, we can see trends that go within the context of different learning rates and baseline models.

In the case of fine-tuning, the model achieved its highest test accuracy (0.8375) with a small learning rate of 0.000100. This aligns with the intuition that a well-trained model requires very little adjustments, and smaller learning rates facilitate fine-tuning without destabilizing the learned features.
Explain – Due to the smaller learning rates during fine-tuning ensure very small model adjustments, while not chainging the learned features. This small steps approach prevents runing over the pre-trained knowledge, and making smooth adaptation to new tasks. For cunclusion - Fine-tuning with smaller rates helps retain valuable patterns while accommodating task-specific nuances.

However in the linear probing model, the highest test accuracy (0.7150) was attained with a larger learning rate of 0.010000. This choice is reasonable as the model necessitates more substantial adaptations in the last layers, and a larger learning rate aids in faster convergence.
Explain - A linear model might need substantial adaptations in its last layers when dealing with more complex or nuanced patterns in the data. The last layers of a linear model are responsible for capturing intricate relationships and features. A larger learning rate facilitates faster convergence by allowing quicker adjustments to these intricate patterns during training. And high learning rates can help the model escape the local minima in the loss landscape, preventing it from getting stuck in suboptimal solutions.
In simpler terms, the higher learning rate helps the linear model adapt more swiftly to the intricate details in the data, improving its ability to understand and represent complex relationships.

In the train-from-scratch model, we cannot discern a clear trend as all accuracies are around 0.5 for the different learning rates. This is likely due because we run only with a single-epoch, making it challenging to observe the impact of learning rates.
Explain - In a single epoch, the model processes the entire training dataset only once, and the learning process may not be sufficient. Making the model struggles to accurately predict labels for all learning rates.

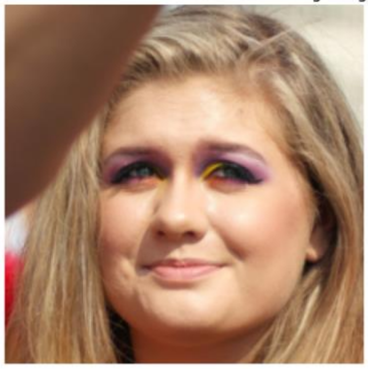
In summary, the optimal learning rates are context-dependent and tied to the specific characteristics of each model.

2. Visualize 5 samples correctly classified by your best baseline but misclas- sified by your worst-performing baselines.

Best model - Loss: 0.4910, Train accuracy: 0.8071, Val accuracy: 0.6350, Test accuracy: 0.6875

Worst model - Loss: 1.4260, Train accuracy: 0.5100, Val accuracy: 0.4900, Test accuracy: 0.4800

Best Model Correct and Worst model Worng Image: 0



Best Model Correct and Worst model Worng Image: 1

Best Model Correct and Worst model Worng Image: 2

Best Model Correct and Worst model Worng Image: 3

Best Model Correct and Worst model Worng Image: 4