A EXPERIMENTATION

The following algorithm is used to run experiments on different datasets given in Section 4.

Algorithm 3 Forward and backward pass of DNN for a single data-point

Given input (x,y), weights (and biases) Θ_l and activations σ_l where $l \in [1, L]$ is the layer and L is the number of layers
$\mathbf{a}_0 = x$;
for $l = 1, \ldots, L$ **do** $h_l = \Theta_l a_{l-1}$;$a_l = \sigma_l(h_l)$
end for
$\dot{a}_L \overset{\text{def}}{=} \nabla_z a_L \leftarrow \frac{\partial \mathcal{L}(z,y)}{\partial z}|_{z=a_L}$
for $l = 1, \ldots, L$ **do** $g_l = \dot{a}_l \bigodot \sigma'(h_l)$;$\dot{\Theta}_l = g_i \hat{a}_{i-1}^\top$;$\dot{a}_{l-1} = \Theta_l^\top g_l$
end for

We present the learning rates for the different algorithms used in Table (4):

Method	Adam	SGD	Adagrad	RMSProp	L-BFGS
Learning rate	1.0×10^{-3}	1.0×10^{-1}	1.0×10^{-2}	1.0×10^{-2}	1.0×10^{0}

Table 4: Learning rates for each update rule.

For the MNIST dataset, the architecture is presented in detail in Table 5. For Fashion-MNIST (FMNIST), the network is described in Table 6. For CIFAR10, the network architecture is described in Table 7. For the autoencoder, we use the architecture presented in Table 8. Finally, the same network architecture is used for reconstruction of FMNIST images.

MNIST network					
Layer	In	Out	Kernel	Stride	Padding
Linear	784	500	-	-	-
ReLU	500	500	-	-	-
Dropout	-	-	-	-	-
Linear	500	10	-	-	-
SoftMax	10	10	-	-	-

Table 5: Network architecture for MNIST.

F-MNIST network					
Layer	In	Out	Kernel	Stride	Padding
Conv2d	1	10	5×5	1	None
MaxPool2d	-	-	2×2	-	-
ReLU	-	-	-	-	-
Conv2d	10	20	5×5	1	None
MaxPool2d	10	10	2×2	-	-
ReLU	-	-	-	-	-
Dropout	-	-	-	-	-
Linear	320	50	-	-	-
Linear	50	10	-	-	-
LogSoftmax	50	10	-	-	-

Table 6: Network architecture for FMNIST.

CIFAR10 network					
Layer	In	Out	Kernel	Stride	Padding
Conv2d	3	6	5×5	1	None
MaxPool2d	-	-	2×2	-	-
ReLU	-	-	-	-	-
Conv2d	6	16	5×5	1	None
MaxPool2d	10	10	2×2	-	-
ReLU	-	-	-	-	-
Linear	120	84	-	-	-
Linear	84	10	-	-	-
LogSoftmax	50	10	-	-	-

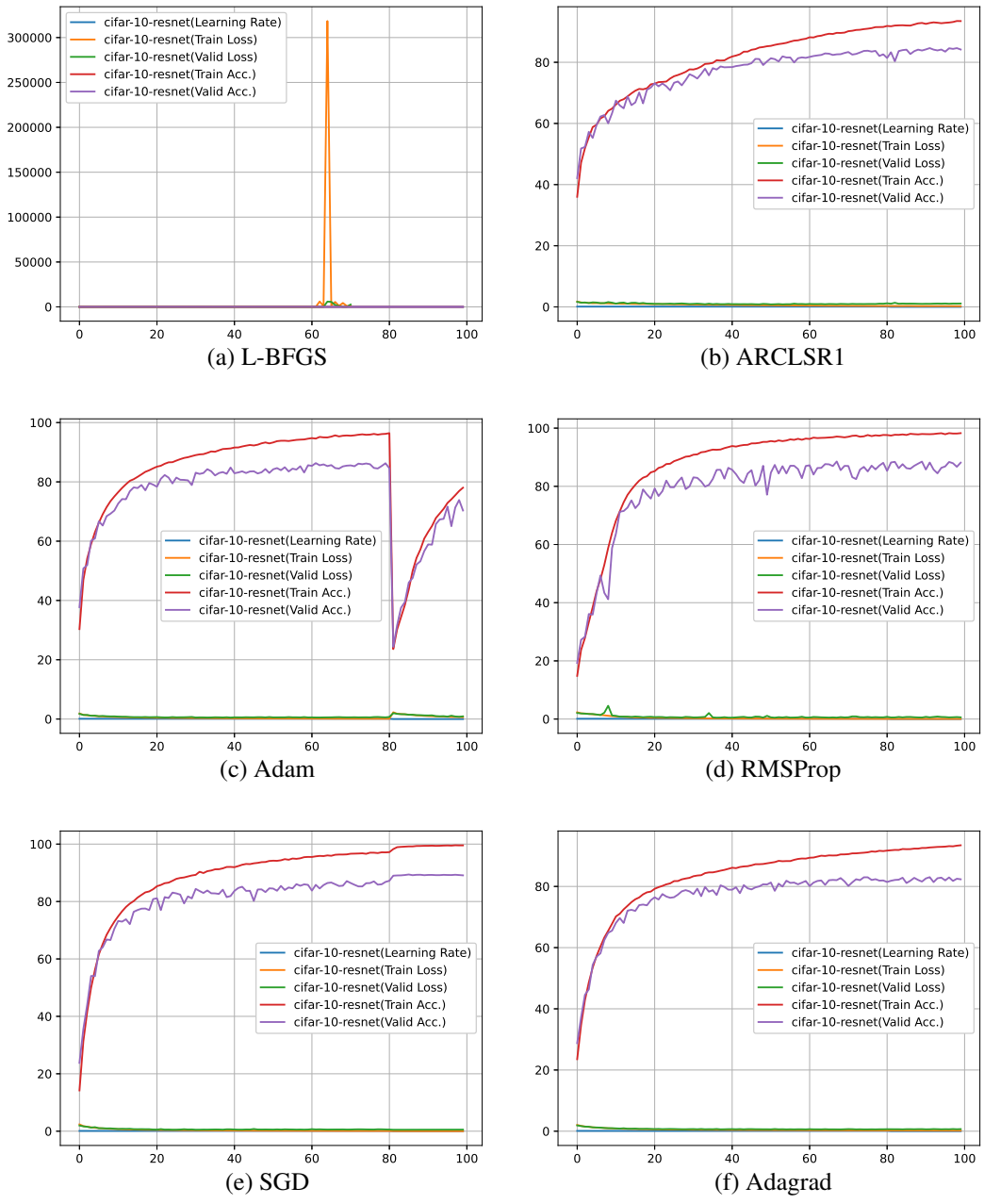
Table 7: Network architecture for CIFAR10.

MNIST-Autoencoder					
Layer	In	Out	Kernel	Stride	Padding
Conv2d	1	8	3×3	2	1
ReLU	-	-	-	-	-
Conv2d	8	16	3×3	2	1
BatchNorm	-	-	-	-	-
ReLU	-	-	-	-	-
Conv2d	16	32	3×3	2	0
Linear	288	128	-	-	-
ReLU	-	-	-	-	-
Linear	128	4	-	-	-
Linear	4	128	-	-	-
ReLU	-	-	-	-	-
Linear	128	288	-	-	-
ReLU	-	-	-	-	-
ConvTrans2d	32	16	3×3	2	0
BatchNorm	-	-	-	-	-
ReLU	-	-	-	-	-
ConvTrans2d	16	8	3×3	2	1
BatchNorm	-	-	-	-	-
ReLU	-	-	-	-	-
ConvTrans2d	8	1	3×3	2	1

Table 8: Network architecture for Fashion-MNIST Autoencoder operation.

B ADDITIONAL RESULTS CIFAR10 USING RESNET50

In Fig. 8, we present the results of classification on CIFAR10 dataset using the ResNet50 architecture. Most methods are able to perform significantly well using the ResNet50 architecture. However, it is interesting to note that most first-order methods begin to over-fit the network on the dataset after roughly 20 epochs. This can be noticed when the validation accuracy deviates from the training accuracy (see Goodfellow et al. (2016)).



(a) L-BFGS (b) ARCLSR1

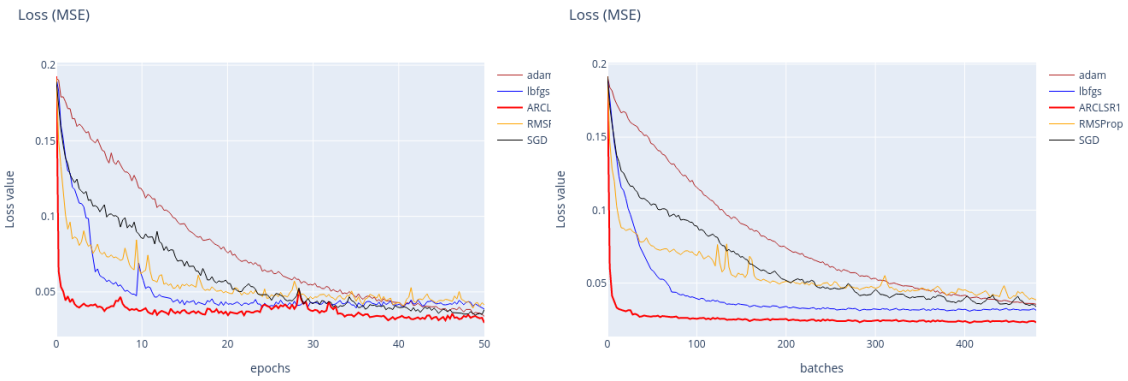
(c) Adam (d) RMSProp

(e) SGD (f) Adagrad

Figure 8: The y-axis represents the classification accuracy on the CIFAR10 dataset, and the x-axis represents the number of epochs. . Each algorithm was run for 100 epochs with a batch-size of 1024 images. In comparison, the ARCLSR-1 training loss tends to be closer to the validation loss, thus preventing over-fitting the network.

C PERFORMANCE ON AUTOENCODER RECONSTRUCTION ON FMNIST

In this sections, we present the training response of the proposed approach on the Fashion-MNIST dataset (Xiao et al. (2017)). We use the same network presented in Table 8.As it can be noticed, similar to the results that we obtained from Section 5, the proposed approach is able to perform better than the adaptive methods in the first half of the first epoch. It comes as no surprise that as the batch size increases, the method performs better.

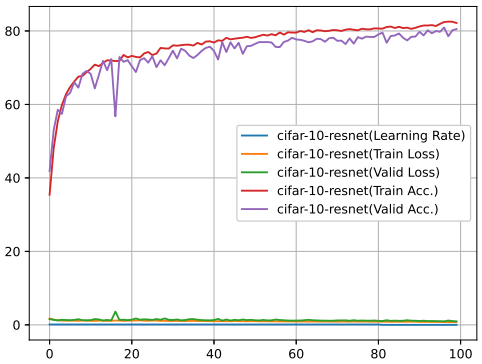


(a) (b)

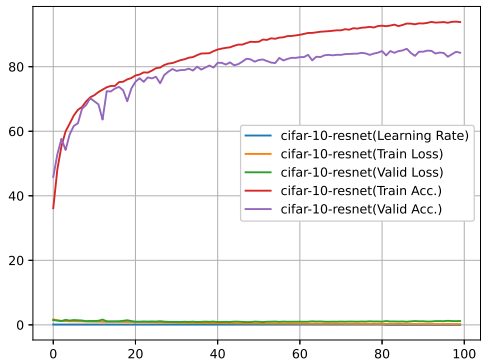
Figure 9: The testing accuracy on F-MNIST dataset. y-axis represents the Mean-Squared Error loss on the training set and x-axis represents the number of epochs/batches. (a) shows the initial training response with a batch size of 256 images. (b) shows the testing response with a batch size of 1000 images.

D PROPOSED APPROACH PERFORMANCE ANALYSIS

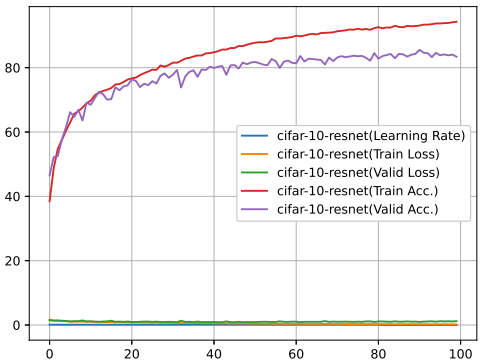
We present the results obtained from using different history sizes and iterations in Figure 10 of the proposed approach on the CIFAR10 dataset using the ResNet50 architecture. **History** here refers to the number of steps ($\mathbf{S} = [\mathbf{s}_0 \dots \mathbf{s}_k]$ and $\mathbf{Y} = [\mathbf{y}_0 \dots \mathbf{y}_k]$ described in Sec. 2) used to approximate the Hessian. The **iter** refers to the number of iterations the proposed approach is operated to minimize the objective function over a batch.



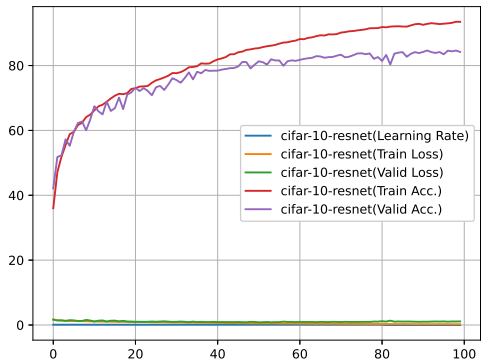
(a) History = 10, iter = 10



(b) History = 20, iter = 20



(c) History = 40, iter = 40



(d) History = 60, iter = 60

Figure 10: The performance of the proposed approach using different variations of history and number of iterations. The ResNet50 architecture was used with a batch size of 1024 images.