Online Resource 2

Datasets, Performance Comparisons, Visualizations, and Evaluation Metrics for the Nova Optimizer

Ali Zeydi Abdian[1], Mohammad Masoud Javidi[2] and Najme Mansouri[3]

[1]Department of Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran, E-mail: ali.zeydi.abdian@ut.ac.ir, alizeydiabdian@math.uk.ac.ir (Corresponding Author)

[2] Professor, Faculty of Shahid Bahonar University of Kerman, Kerman, Iran, Email: javidi@uk.ac.ir

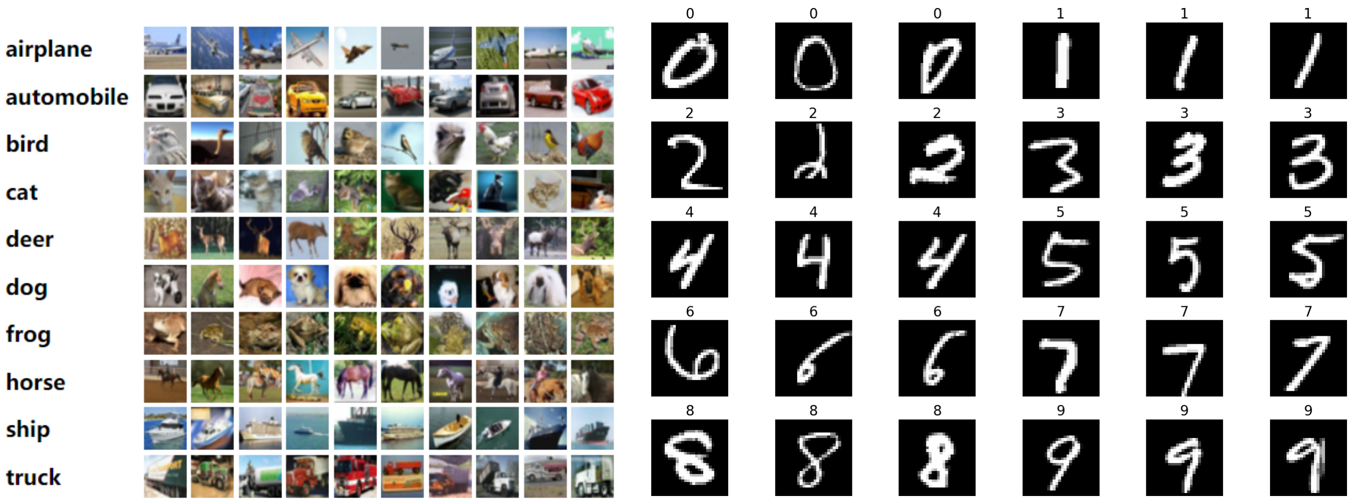
[3] Associate Professor, Faculty of Shahid Bahonar University of Kerman, Kerman, Iran, Email: n.mansouri@uk.ac.ir, najme.mansouri@gmail.com, Box No. 76135-133, Fax: 03412111865

Corresponding author: alizeydiabdian@math.uk.ac.ir

Journal Name: Signal, Image and Vide Processing

Contents

1 Datasets



(a) CIFAR-10 (b) MNIST

Fig 1. Visualization of CIFAR-10 and MNIST datasets.

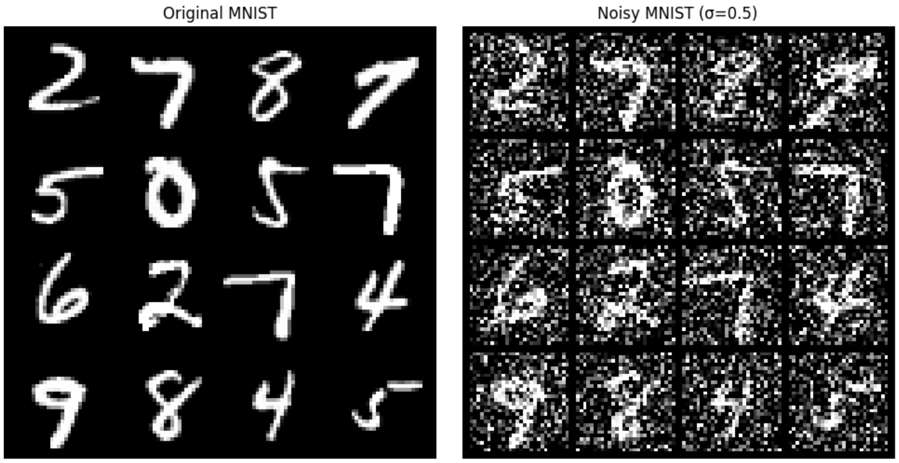


Fig 2. MNIST and noisy MNIST

2 Time and Space Complexity of Common First-Order Optimizers

Let P denote the total number of trainable parameters in a model. Each optimizer updates all parameters in one pass, performing a constant number of element-wise operations per parameter, so the *time complexity* per iteration is $\mathcal{O}(P)$.

Many optimizers maintain auxiliary state vectors of length P. If an optimizer keeps k such vectors, its *additional memory complexity* is:

$$\mathcal{O}(\mathrm{kP}) \quad \text{(beyond the P parameters and their gradients).}$$

Table 1 summarizes the relative computational efficiency and memory usage of various optimizers

Table 1. Comparison of time and space complexity for popular gradient-based optimizers. Here P is the number of model parameters; "Memory overhead" counts only the auxiliary state beyond parameters and gradients.

Optimizer	Time per update	Memory overhead	State vectors per parameter
SGD	$\mathcal{O}(\mathrm{P})$	$\mathcal{O}(0 \cdot \mathrm{P})$	None
SGD + Momentum	$\mathcal{O}(\mathrm{P})$	$\mathcal{O}(1 \cdot \mathrm{P})$	1 (velocity)
RMSprop	$\mathcal{O}(\mathrm{P})$	$\mathcal{O}(1 \cdot \mathrm{P})$	1 (running $\mathbb{E}[\mathrm{g}^2]$)
Adam / AdamW	$\mathcal{O}(\mathrm{P})$	$\mathcal{O}(2 \cdot \mathrm{P})$	2 (first and second moments: m, v)
NAdam	$\mathcal{O}(\mathrm{P})$	$\mathcal{O}(2 \cdot \mathrm{P})$	2 (as in Adam)
Nova (AMSGrad-style)	$\mathcal{O}(\mathrm{P})$	$\mathcal{O}(3 \cdot \mathrm{P})$	3 (m, v, and $\max(\mathrm{v})$)

3 Evaluating the Performance of Various Optimizers on Some Well-Known Networks: A Comparison with Nova

Table 2. ResNet-18 architecture details

Layer / Block	Operation	Output / Notes
Input	RGB Image	224×224×3
Conv1	7×7 Convolution, 64 filters, stride 2, padding 3	112×112×64
BN + ReLU	–	–
MaxPool	3×3 Max Pooling, stride 2	56×56×64
Residual Blocks (each block: 2 conv layers, BN, ReLU)		
Layer 1 (conv2_x)	2 blocks with 64 filters (stride 1)	56×56×64
Layer 2 (conv3_x)	2 blocks with 128 filters; first block uses stride 2	28×28×128
Layer 3 (conv4_x)	2 blocks with 256 filters; first block uses stride 2	14×14×256
Layer 4 (conv5_x)	2 blocks with 512 filters; first block uses stride 2	7×7×512
Global Average Pooling	Pooling over 7×7 spatial dimensions	1×1×512
FC Layer	Fully Connected, e.g. 1000 classes for ImageNet	1000 (or as specified)

Table 3. LeNet-5 architecture details

Layer	Operation	Output / Notes
Input	Grayscale image	32×32×1
C1	Convolution (5×5 kernel), 6 filters, activation tanh	28×28×6
S2	Average pooling (subsampling)	14×14×6
C3	Convolution (5×5 kernel), 16 filters, tanh	10×10×16
S4	Average pooling (subsampling)	5×5×16
C5	Convolution (fully-connected), kernel size matching input, tanh	1×1×120
F6	Fully connected	84 units
Output	Fully connected	10 units (for digit classification)

Table 4. Comparison of ResNet-18 and LeNet-5

Attribute	**ResNet-18**	**LeNet-5**
Year developed	2015	1998
Depth (Layers)	18	7
Key Innovation	Residual (skip) connections	Convolution + pooling
Original application	Large-scale image recognition	Handwritten digit recognition
Training techniques	Batch normalization, deeper architecture	Simpler activations (e.g., tanh)
Network complexity	High (designed for complex datasets)	Low (optimized for limited variability)

Table 5. Used architeture CIFAR-10 ResNet-18 in the paper

Layer/Block	**Modification**	**Parameters/Notes**
Input	CIFAR-10 Image	32×32×3
Conv1 MaxPool	3×3 Conv, stride 1 Removed	Original: 7×7 stride 2 Identity operation
Residual Blocks	Added Spatial Dropout	Layer1: 0.05, Layer2: 0.1 Layer3: 0.15, Layer4: 0.2
Classifier	Modified FC Layers Added Dropout	512 → 256 → 10 FC dropout: 0.3

Table 6. Used architeture MNIST LeNet-5 in the paper

Layer	**Operation**	**Parameters**
Input	MNIST Image	28×28×1
Conv1 MaxPool	5×5 Conv, 6 filters 2×2	Output: 24×24×6 Output: 12×12×6
Conv2 MaxPool	5×5 Conv, 16 filters 2×2	Output: 8×8×16 Output: 4×4×16
FC1 FC2 FC3	Linear Layer Linear Layer Linear Layer	256 → 120 120 → 84 84 → 10
Regularization	Dropout	p=0.25 after FC1

Table 7. Training configuration

Category	**Parameter**	**Value**
Training	Epochs	30
	Batch Size	128
	LR Scheduler	StepLR (gamma=0.9)
Optimizers	Base LR	0.001
	Weight Decay	0.01
	Nova	betas=(0.9, 0.999), eps=1e-8
	SGD	momentum=0.9
Data Augmentation	CIFAR-10	RandomCrop(32,4)+Flip
	MNIST	None
Normalization	CIFAR-10	Mean=[0.4914, 0.4822, 0.4465]
		Std=[0.2023, 0.1994, 0.2010]
	MNIST	Mean=0.1307, Std=0.3081

Table 8. Training configuration for SST-2

Category	**Parameter**	**Value**
Training	Epochs	10
	Batch Size	64
	LR Scheduler	StepLR (gamma=0.9)
Optimizers	Base LR	0.001
	Weight Decay	0.0
	Nova	betas=(0.9, 0.999), eps=1e-8
	Adam	Default parameters
Data Augmentation	SST-2	None
Normalization	Preprocessing	Lowercase conversion, tokenization, and padding
	Vocabulary Construction	Count-based, with ‘`<unk>`‘ and ‘`<pad>`‘ tokens

Table 9. Detailed architecture of SST-2 text classification model

Component	**Description**
Model Type	Feedforward Neural Network (Embedding + Linear Classifier)
Embedding Layer	• Input dimension: vocab_size (determined from training data) • Output dimension (embed_dim): 100 (configurable) • Special tokens: `<unk>`=0, `<pad>`=1 (for OOV words and padding) • Trainable parameters: vocab_size × embed_dim
Pooling	• Operation: Mean pooling across sequence length (dim=1) • Reduces variable-length sequences to fixed-size embedding
Classifier Layer	• Single Linear layer (embed_dim → num_class) • Output size: 2 (binary classification: Negative/Positive) • No explicit activation (CrossEntropyLoss includes LogSoftmax) • Trainable parameters: embed_dim × num_class + bias
Forward Pass	1. Tokenize text → convert to vocab indices 2. Embedding layer: maps indices to dense vectors 3. Mean pooling: aggregates sequence embeddings 4. Linear layer: produces logits for classification
Training Hyperparameters	• Batch size: 64 • Loss: CrossEntropyLoss (combines LogSoftmax + NLLLoss) • Optimizer: Configurable (e.g., Adam, SGD) • Learning rate scheduler: StepLR (step_size=1, gamma=0.9)
Data Pipeline	• Dataset: SST-2 (Stanford Sentiment Treebank v2) • Splits: 90% train / 10% val (from original train), test=validation set • Preprocessing: Lowercase + NLTK word_tokenize • Vocabulary: Built from training set, with `<unk>` and `<pad>` • Padding: Dynamic per batch (to max length in batch)

4 Confusion Matrix

Table 10. Confusion matrix and evaluation metrics

		Predicted class	
		Positive	**Negative**
Actual class	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)
		$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$	$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$
		$\text{Accuracy} = \frac{\text{TN+TP}}{\text{TP+TN+FP+FN}}$	$\text{F1 score} = \frac{\text{2TP}}{\text{2TP+FP+FN}}$

5 Results

5.1 Comparison of Performance Metrics of Nova with Other Optimizers

Table 11. Performance metrics on CIFAR-10

Optimizer	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Test Loss	Test Acc (%)	Epoch Time (s)
Nova	0.1550	94.70	0.3180	89.42	0.3345	90.01	41.95
Adam	0.4788	84.94	0.5094	83.08	0.5069	83.14	39.99
RMSprop	1.1128	59.39	1.0714	60.36	1.0597	61.01	38.50
SGD	1.2252	55.11	1.1721	57.88	1.1325	58.57	39.87
Nadam	0.5583	82.09	0.5642	81.20	0.5718	80.85	40.21

Table 12. Classification metrics on CIFAR-10

Optimizer	Precision (%)	Recall (%)	F1-score (%)
Nova	89.98	90.01	89.99
Adam	83.21	83.14	83.00
RMSprop	61.01	61.01	59.77
SGD	58.89	58.57	58.18
Nadam	80.93	80.85	80.82

Table 13. Performance metrics on MNIST

Optimizer	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Test Loss	Test Acc (%)	Epoch Time (s)
Nova	0.0204	99.33	0.0370	98.86	0.0295	98.95	12.01
Adam	0.0673	98.20	0.0578	98.10	0.0504	98.58	11.22
RMSprop	0.0699	98.08	0.0631	97.94	0.0514	98.50	11.21
SGD	0.1240	96.45	0.0960	97.12	0.0855	97.51	11.16
Nadam	0.0661	98.21	0.0582	98.16	0.0484	98.66	11.34

Table 14. Classification metrics on MNIST

Optimizer	Precision (%)	Recall (%)	F1-score (%)
Nova	98.95	98.94	98.94
Adam	98.58	98.57	98.57
RMSprop	98.50	98.48	98.49
SGD	97.52	97.49	97.50
Nadam	98.67	98.65	98.66

Table 15. Performance metrics for SST-2

Optimizer	Test Loss	Test Acc (%)	Precision (%)	Recall (%)	F1-score (%)	Epoch Time (s)
Nova	0.5125	82.00	82.00	82.01	81.99	15.48
Adam	0.6397	66.28	68.71	65.94	64.85	15.31
AdamW	0.5804	81.65	81.69	81.62	81.63	15.28
RMS	0.6422	65.94	69.05	65.55	64.15	15.23
SGD	0.6810	54.47	56.64	53.88	48.85	15.08
Nadam	0.6399	66.17	68.84	65.81	64.61	15.36

Table 16. Classification metrics on SST-2

Optimizer	Precision (%)	Recall (%)	F1-score (%)
Nova	82.11	82.12	82.11
Adam	69.56	66.38	65.19
AdamW	80.93	80.39	80.39
RMS	68.10	65.48	64.41
SGD	59.31	53.28	44.63
Nadam	69.62	66.13	64.81

Table 17. Performance metrics for noisy MNIST

Optimizer	Test Loss	Test Acc (%)	Precision (%)	Recall (%)	F1-score (%)	Epoch Time (s)
Nova	0.1065	96.58	96.56	96.56	96.56	8.47
AdamW	0.1112	96.14	96.14	96.11	96.12	7.85
Adam	0.1156	96.17	96.16	96.15	96.14	7.77
RMS	0.1072	96.53	96.52	96.50	96.50	7.75
SGD	0.2727	91.32	91.24	91.21	91.20	7.70
Nadam	0.1052	96.43	96.42	96.39	96.40	7.63

Table 18. Performance of Nova optimizer hyperparameter combinations on MNIST

Combination	LR	β_1	β_2	WD	Test Acc (%)	Precision (%)	Recall (%)	F1-score (%)
Comb1	0.001	0.9	0.999	0.01	99.21	99.21	99.20	99.20
Comb2	0.001	0.9	0.99	0.01	99.17	99.16	99.16	99.16
Comb3	0.001	0.95	0.999	0.001	99.11	99.11	99.10	99.11
Comb4	0.001	0.95	0.99	0.001	98.94	98.94	98.93	98.93
Comb5	0.01	0.9	0.999	0.01	99.14	99.14	99.13	99.13
Comb6	0.01	0.9	0.99	0.01	99.01	99.01	98.99	99.00
Comb7	0.01	0.95	0.999	0.001	98.44	98.44	98.42	98.43
Comb8	0.01	0.95	0.99	0.001	99.00	99.00	98.99	98.99

5.2 Learning Curves and Bar Plots

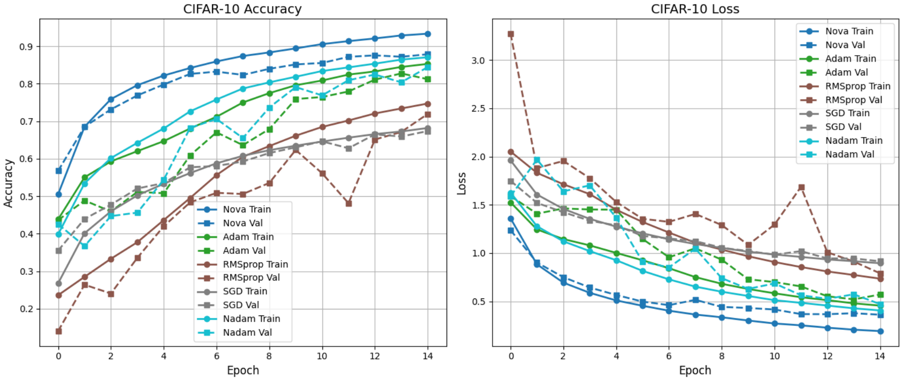


Fig 3. Learning curve CIFAR-10 with different optimizers

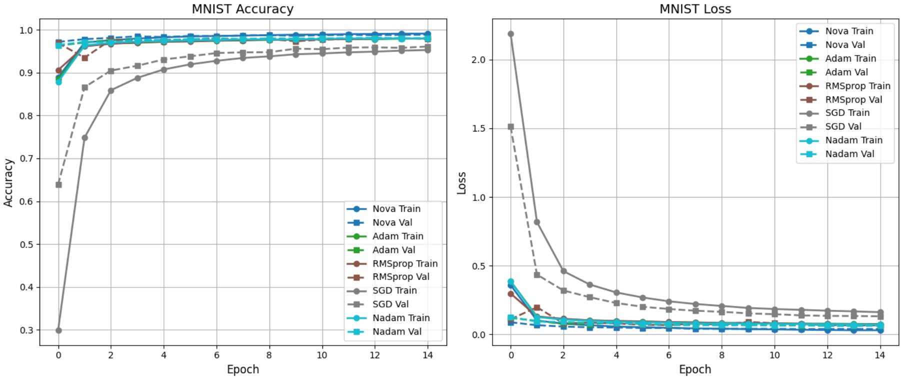


Fig 4. Learning curve MNIST with different optimizers

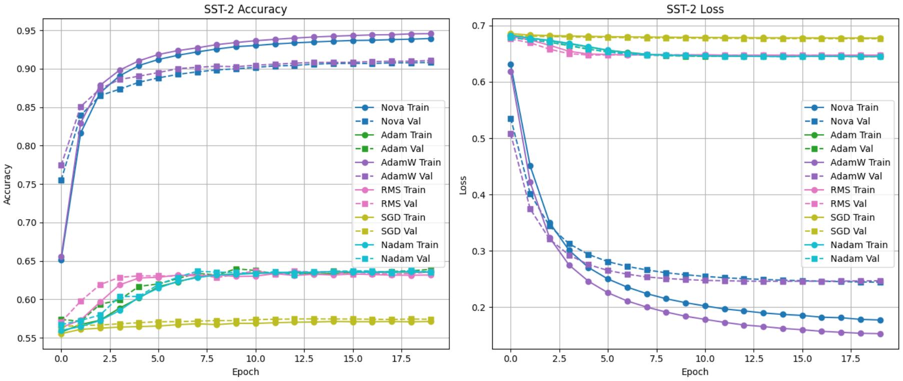


Fig 5. Learning curve SST-2 with different optimizers

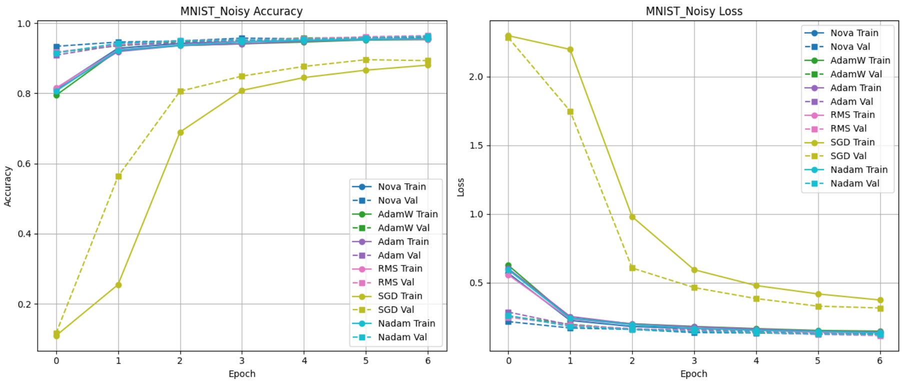


Fig 6. Learning curve Noisy MNIST with different optimizers

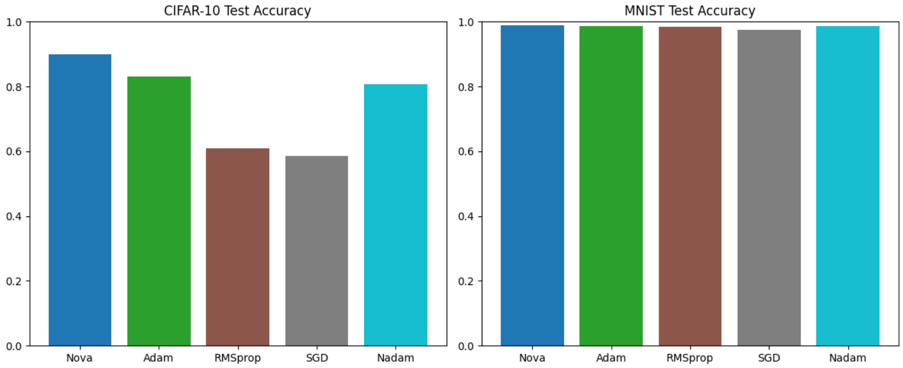
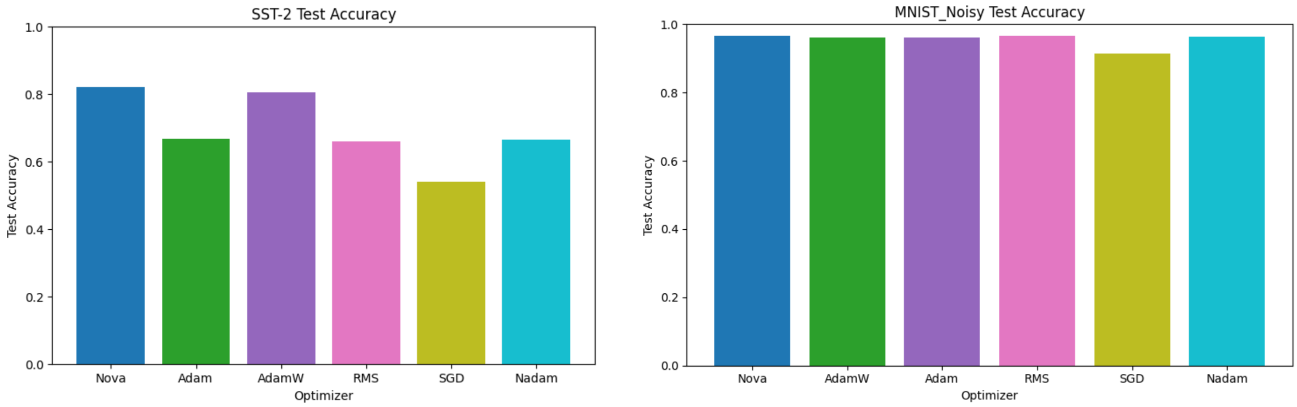


Fig 7. Bar plot for comparisoning accuracy of Nova with other optimizers (CIFAR-10 and MNIST)

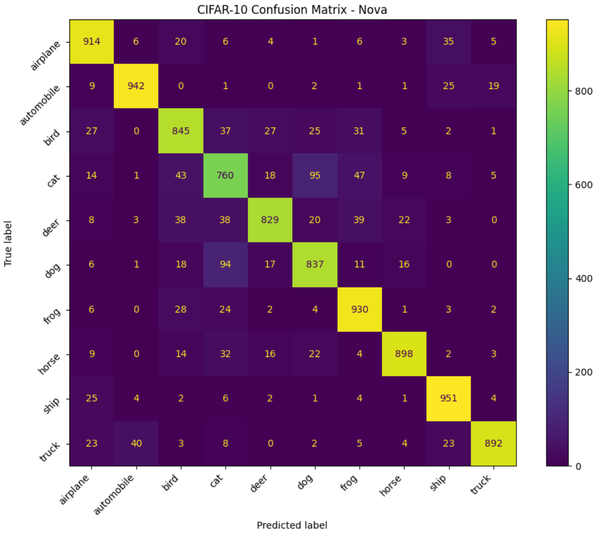


(a) Bar plot for comparisoning accuracy of Nova with other optimizers (SST-2)

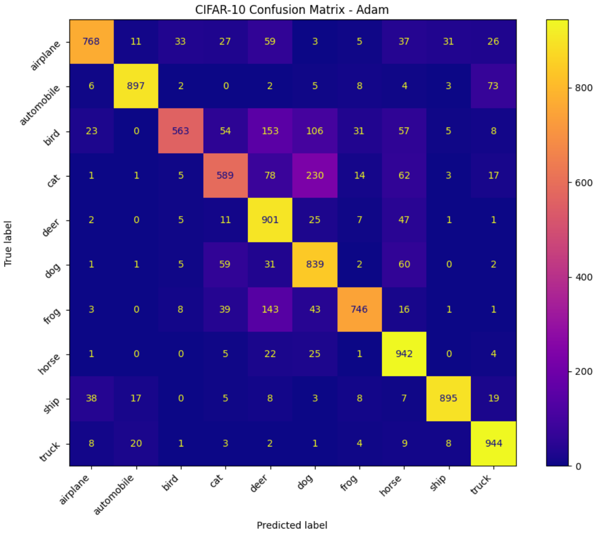
(b) Bar plot for comparisoning accuracy of Nova with other optimizers (noisy MNIST)

Fig 8. Bar plot for comparisoning accuracy of Nova with other optimizers

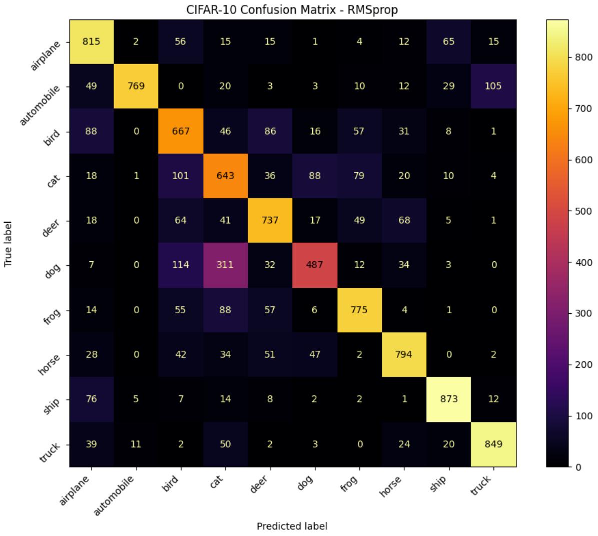
5.3 Confusion Matrices of datasets of CIFAR-10, MNIST, Noisy MNIST and SST-2



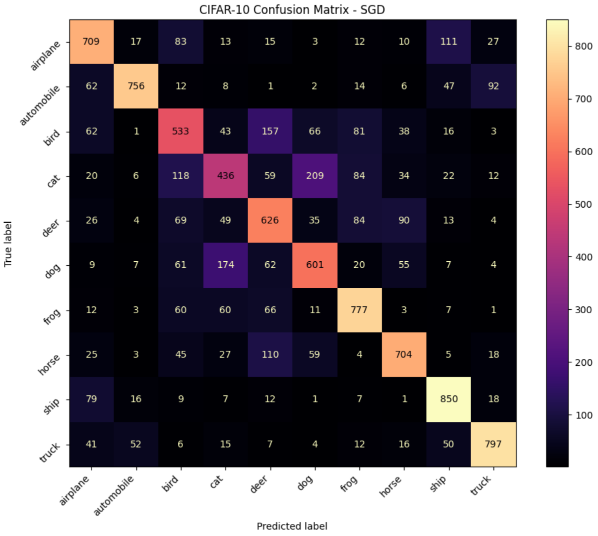
(a) Confusion matrix of CIFAR-10 (Nova)



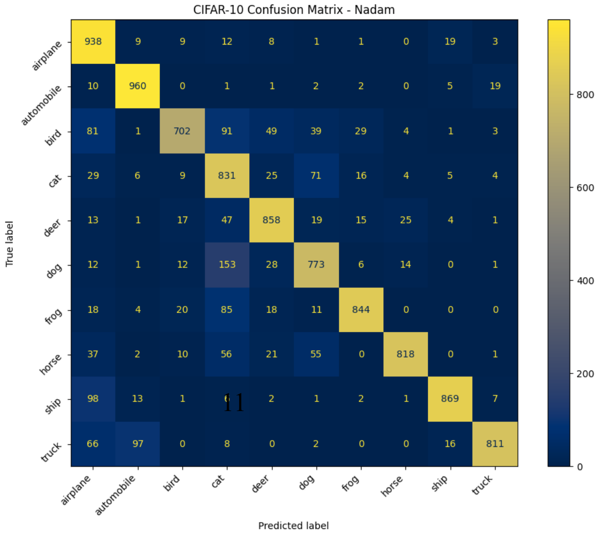
(b) Confusion matrix of CIFAR-10 (Adam)



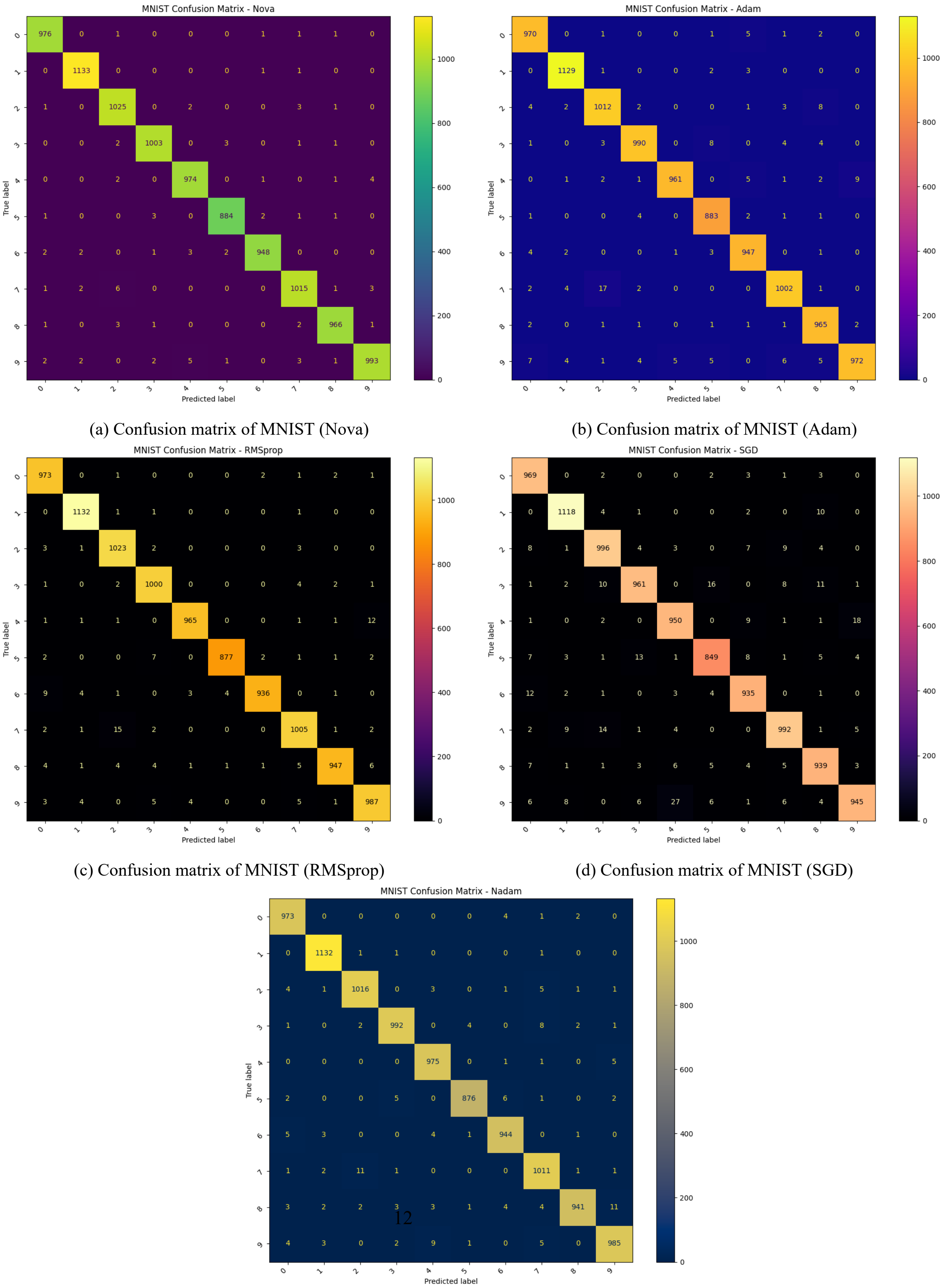
(c) Confusion matrix of CIFAR-10 (RMSprop)



(d) Confusion matrix of CIFAR-10 (SGD)



(e) Confusion matrix of CIFAR-10 (Nadam)



(a) Confusion matrix of MNIST (Nova)

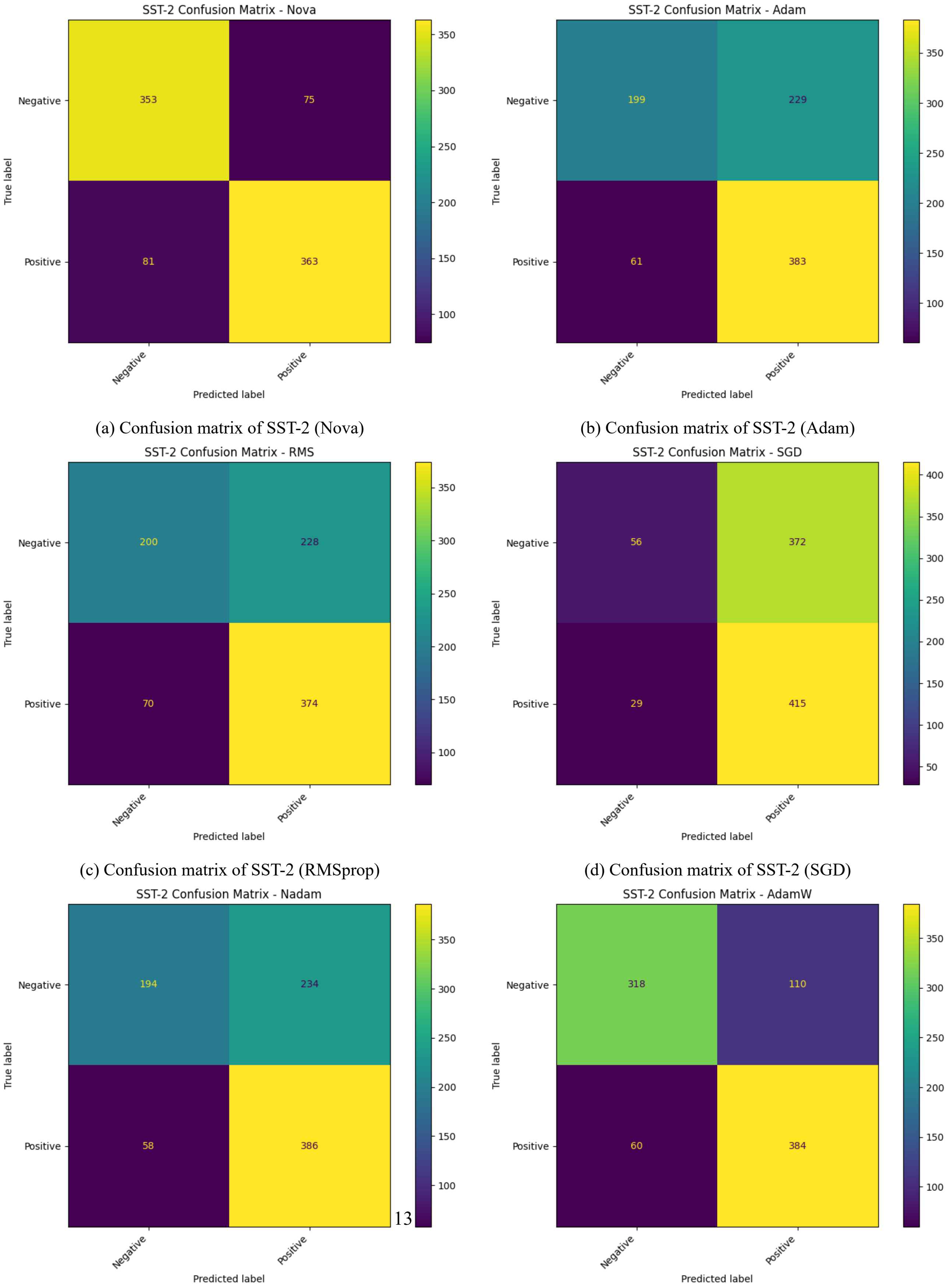
(b) Confusion matrix of MNIST (Adam)

(c) Confusion matrix of MNIST (RMSprop)

(d) Confusion matrix of MNIST (SGD)

(e) Confusion matrix of MNIST (Nadam)

Fig 10. Confusion matrix for dataset with respect to different optimizers



(a) Confusion matrix of SST-2 (Nova)

(b) Confusion matrix of SST-2 (Adam)

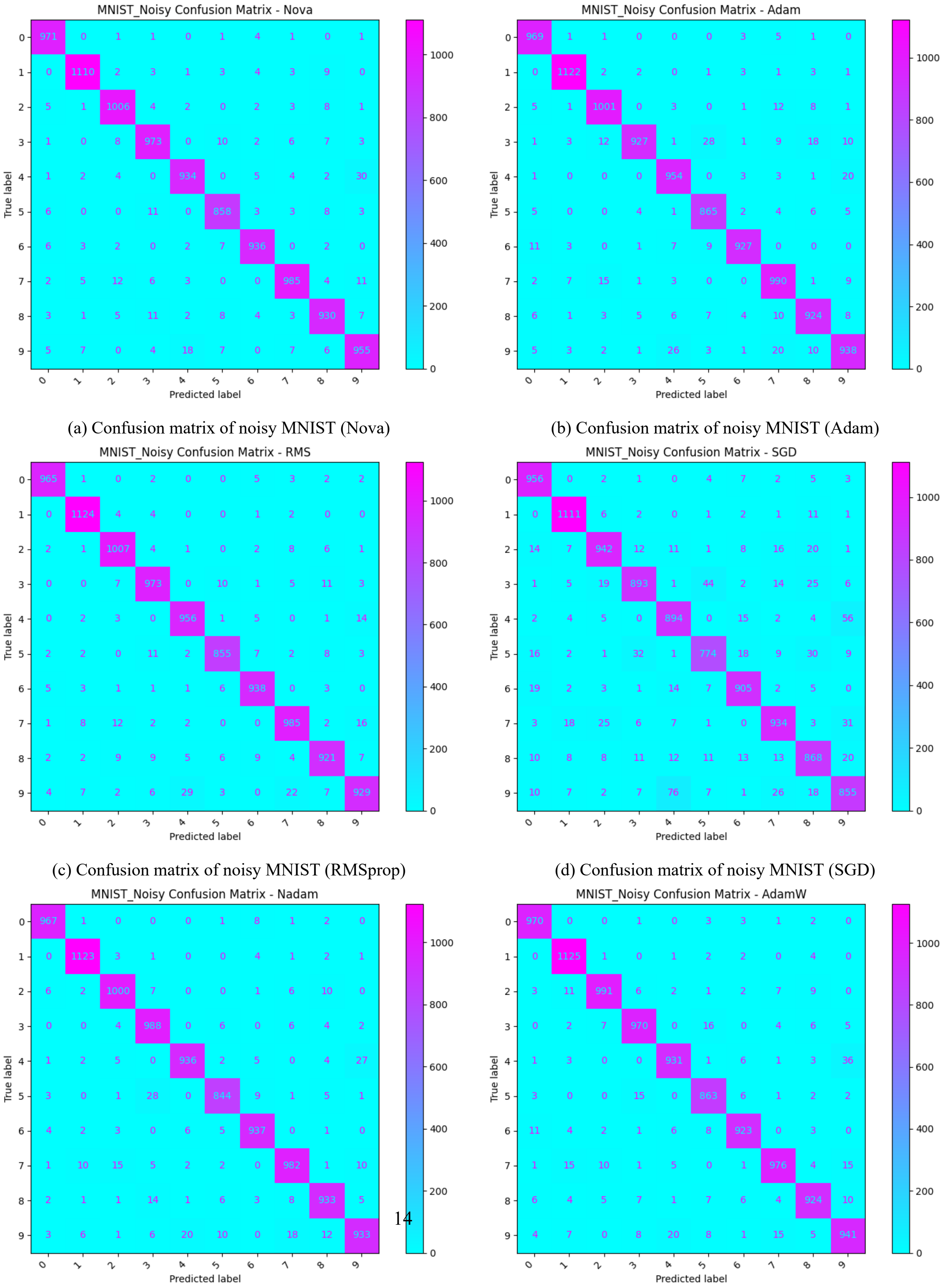
(c) Confusion matrix of SST-2 (RMSprop)

(d) Confusion matrix of SST-2 (SGD)

(e) Confusion matrix of SST-2 (Nadam)

(f) Confusion matrix of SST-2 (AdamW)

Fig 11. Confusion matrix for dataset with respect to different optimizers



(a) Confusion matrix of noisy MNIST (Nova)

(b) Confusion matrix of noisy MNIST (Adam)

(c) Confusion matrix of noisy MNIST (RMSprop)

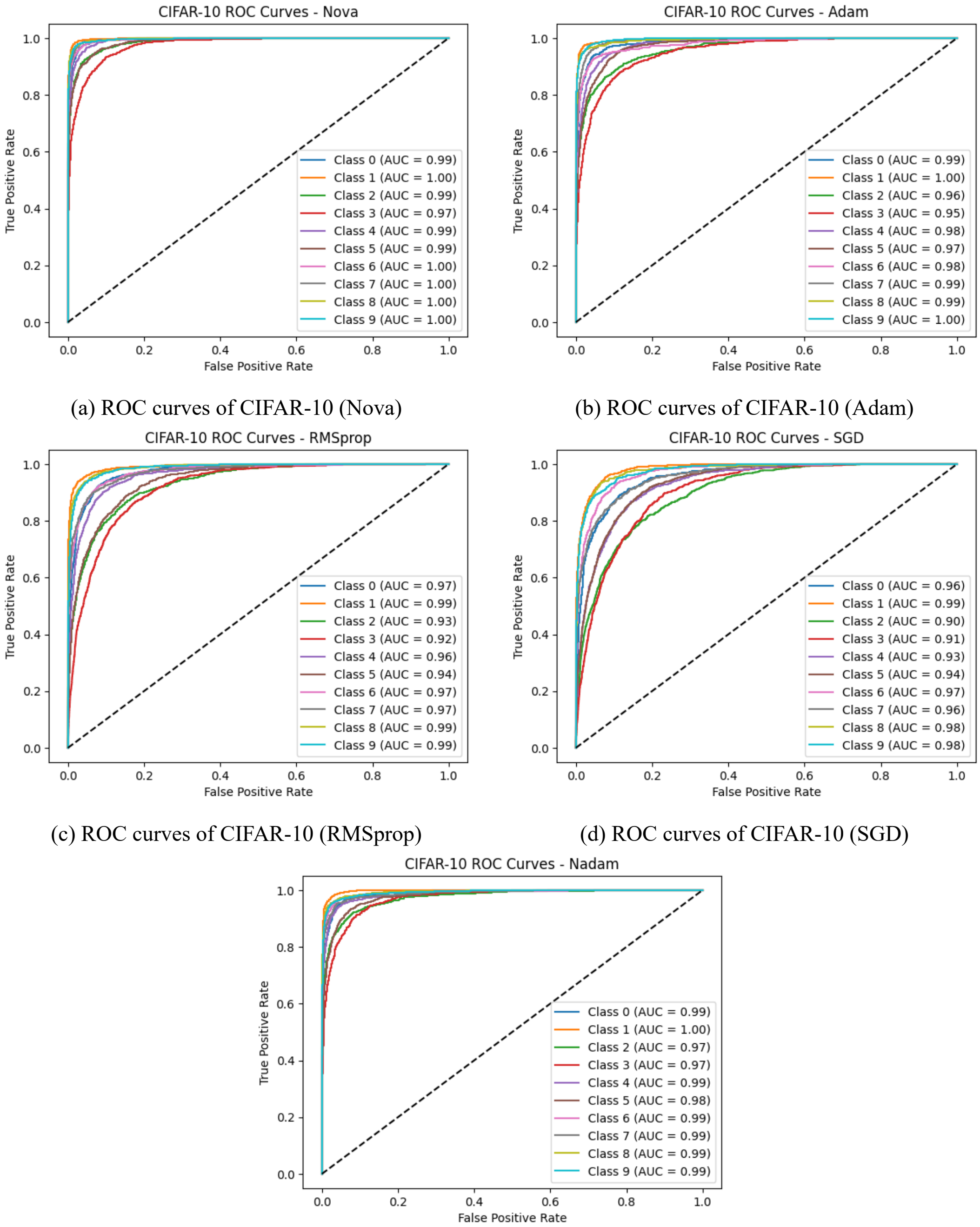
(d) Confusion matrix of noisy MNIST (SGD)

(e) Confusion matrix of noisy MNIST (Nadam)

(f) Confusion matrix of noisy MNIST (AdamW)

Fig 12. Confusion matrix for dataset with respect to different optimizers

5.4 ROC and AUC



(a) ROC curves of CIFAR-10 (Nova)

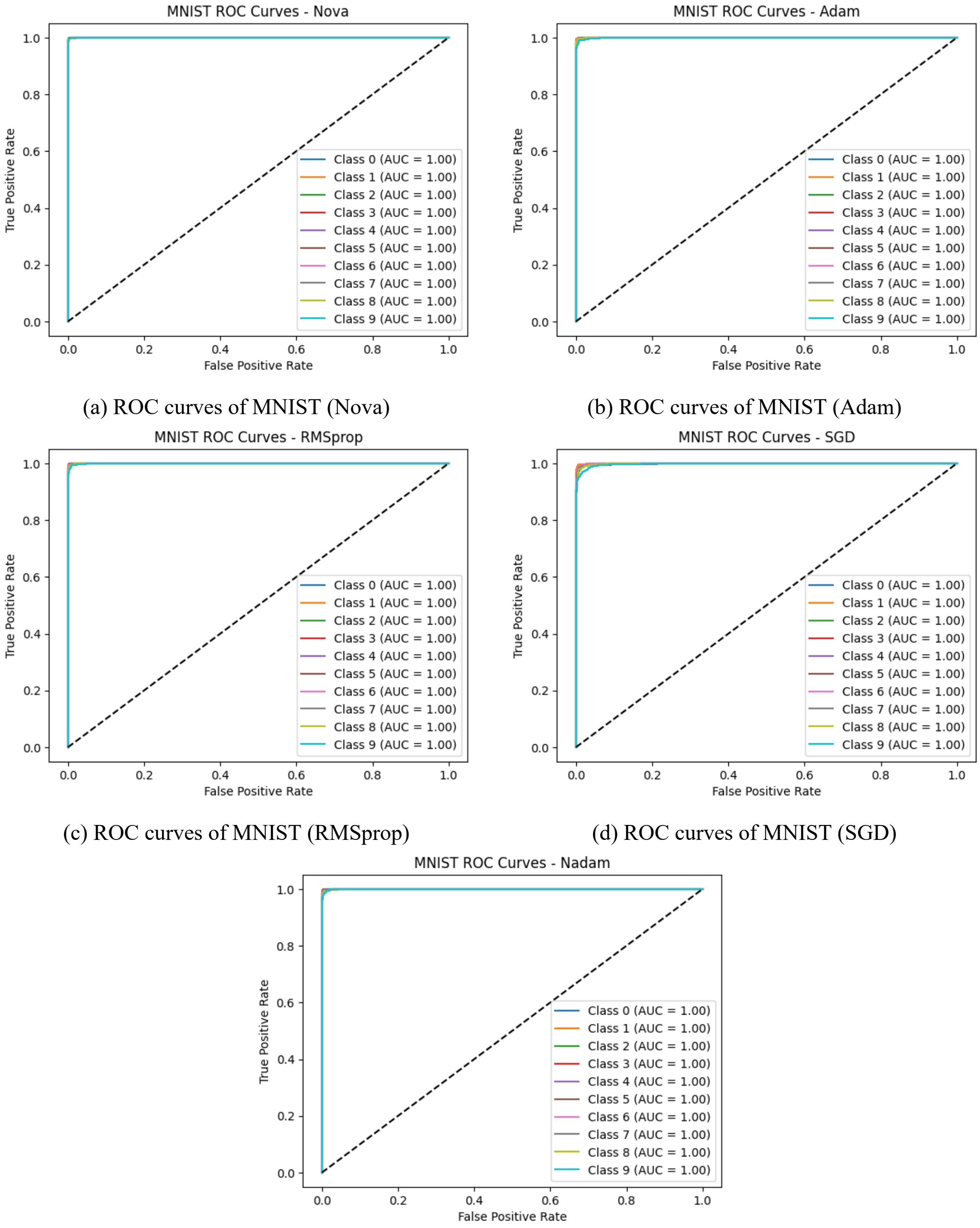
(b) ROC curves of CIFAR-10 (Adam)

(c) ROC curves of CIFAR-10 (RMSprop)

(d) ROC curves of CIFAR-10 (SGD)

(e) ROC curves of CIFAR-10 (Nadam)

Fig 13. ROC curves for CIFAR-10 dataset with respect to different optimizers



(a) ROC curves of MNIST (Nova)

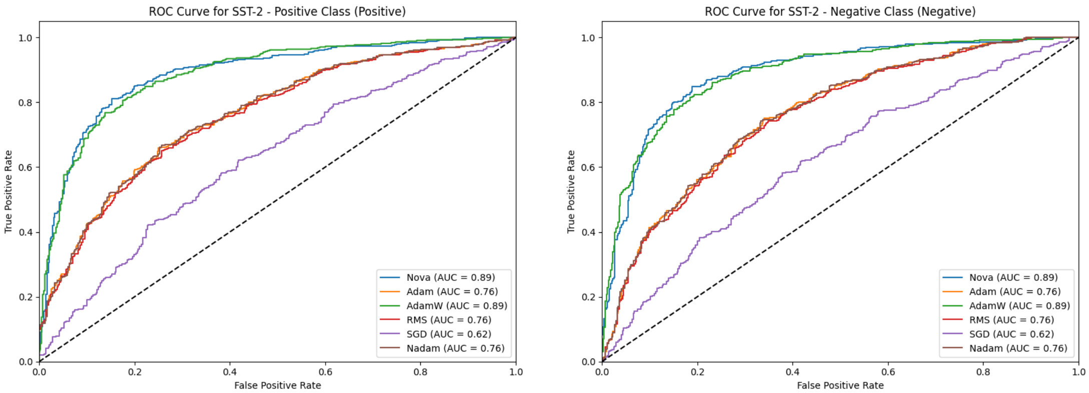
(b) ROC curves of MNIST (Adam)

(c) ROC curves of MNIST (RMSprop)

(d) ROC curves of MNIST (SGD)

(e) ROC curves of MNIST (Nadam)

Fig 14. ROC curves for MNIST dataset with respect to different optimizers



(a) ROC of SST-2 (Positive class)

(b) ROC of SST-2 (Negative class)

Fig 15. ROC curves for SST-2 dataset with respect to different optimizers