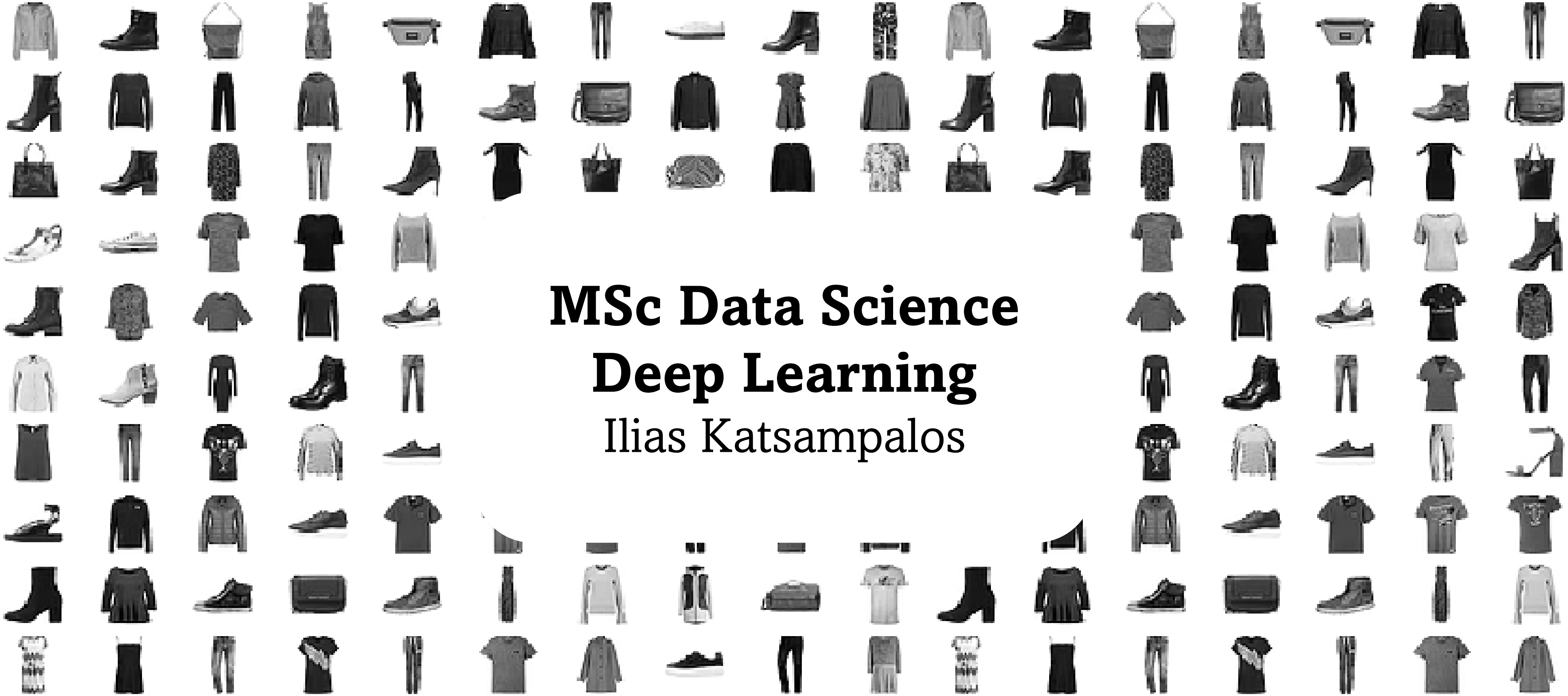
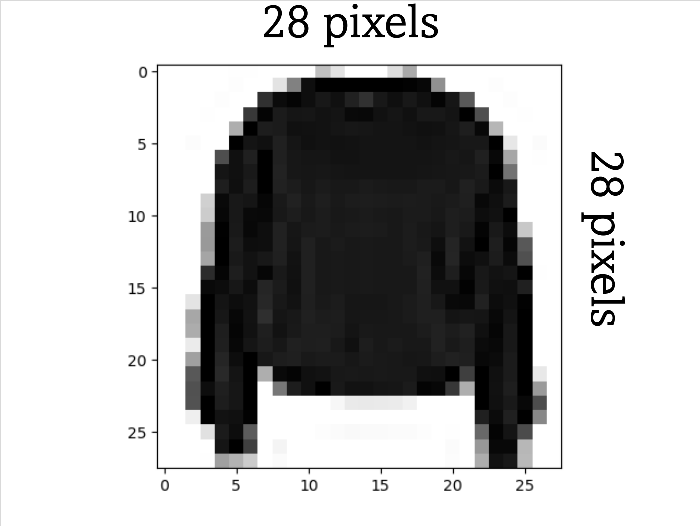


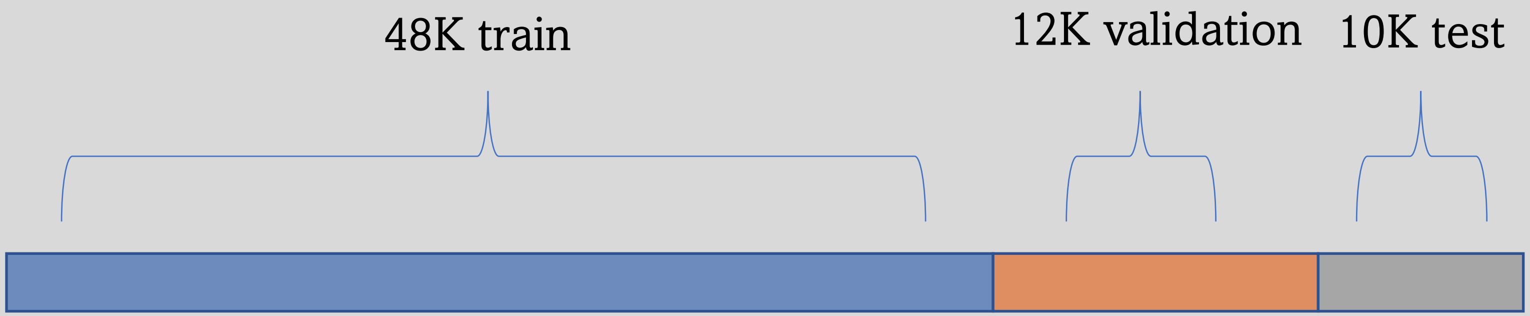
MSc Data Science
Deep Learning

Ilias Katsampalos

Fashion MNIST

- 70 K Images of clothing items
- Greyscale, only 1 channel
- 10 labels (shoes, shirts etc.)

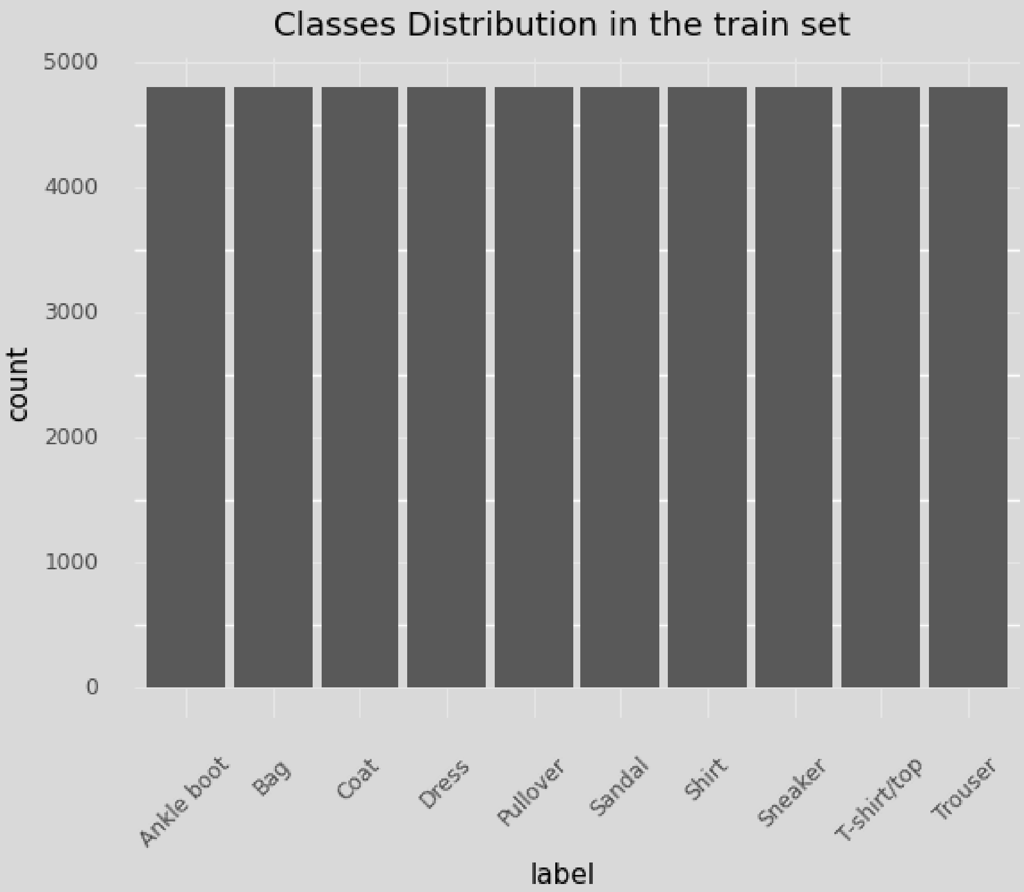


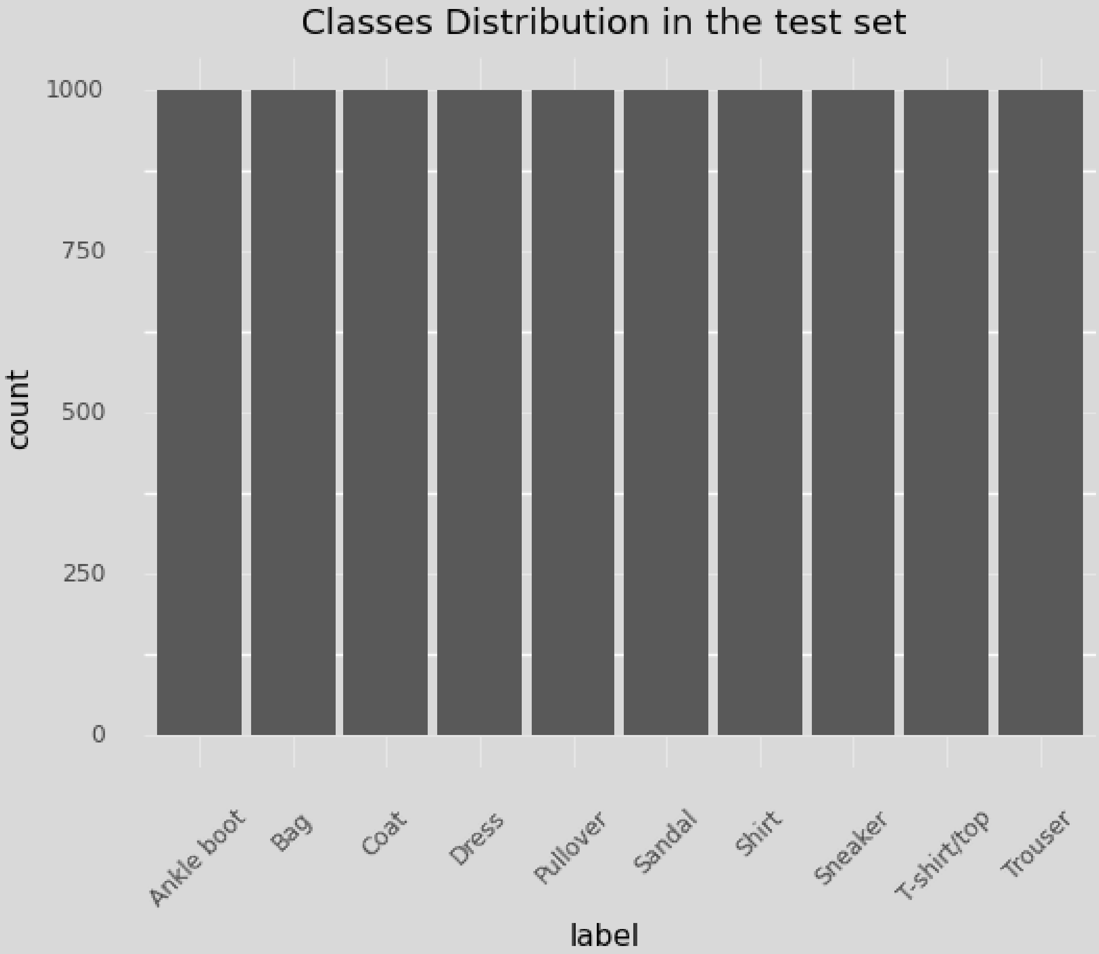

48K train

12K validation

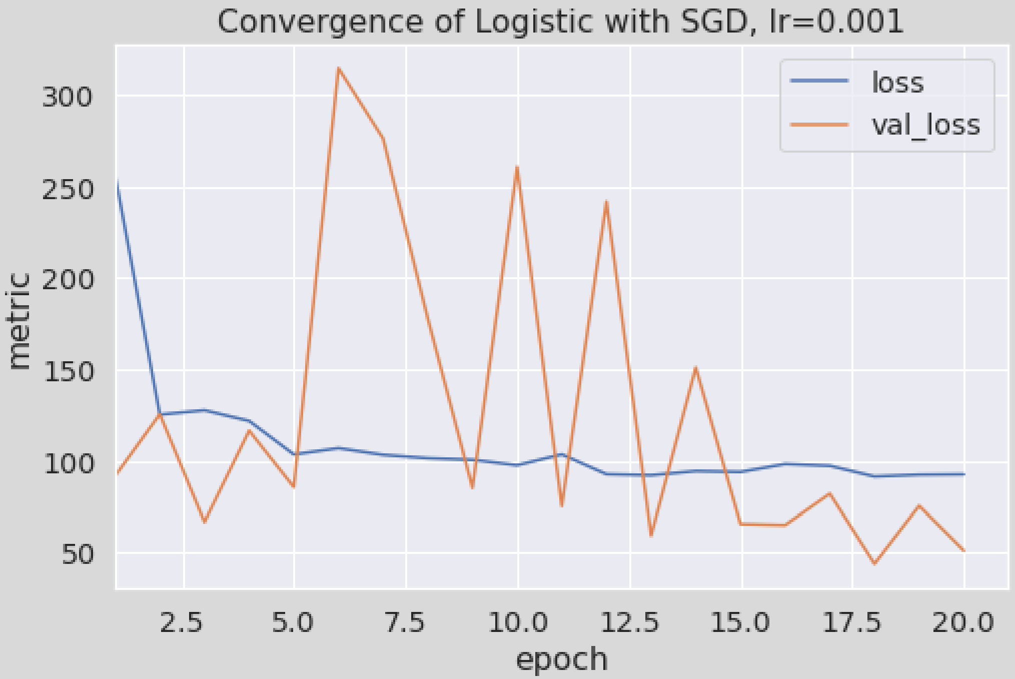
10K test

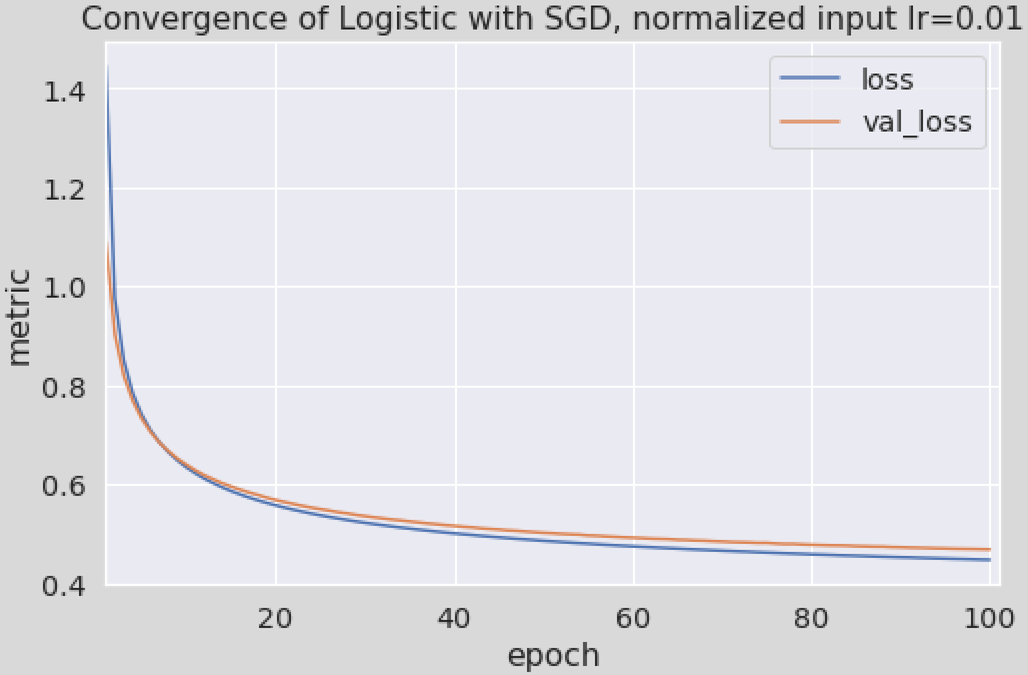
Class Distribution – Absolutely Balanced!


Classes Distribution in the train set


Classes Distribution in the test set

Shallow networks & the importance of input standardization



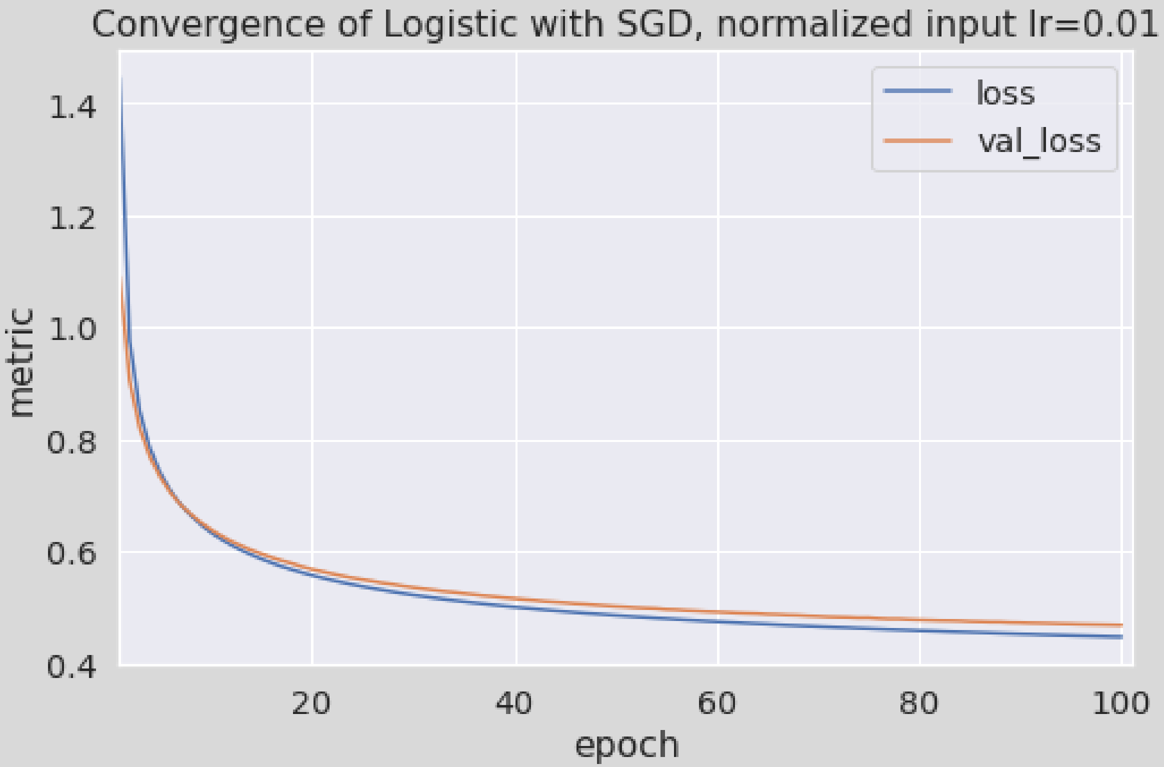


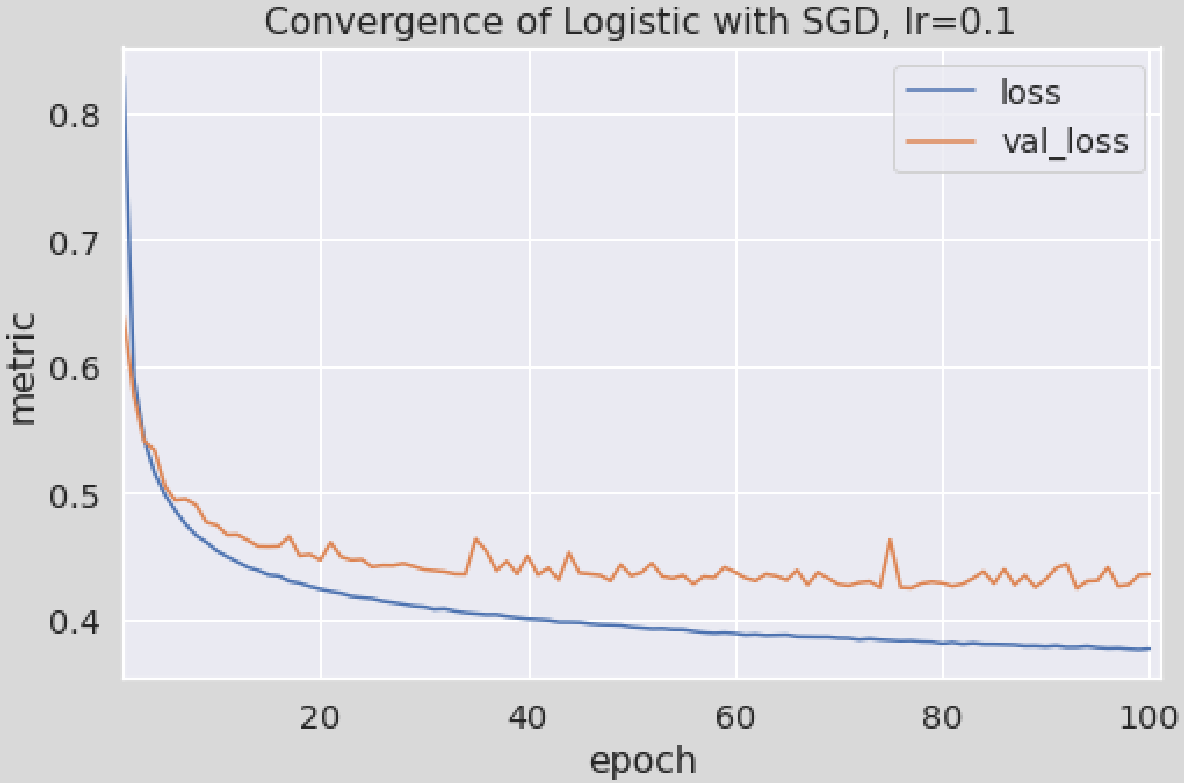
- Shallow network with SGD
- Pixels range from 0 to 255
- Big fluctuations in the validation loss
- Large magnitudes cause exploding gradients
- Weights are updated to meaningless spaces

standardize →

- Pixels range from 0 to 1
- Better and smoother convergence
- Need to increase the learning rate

Shallow networks - A good learning rate



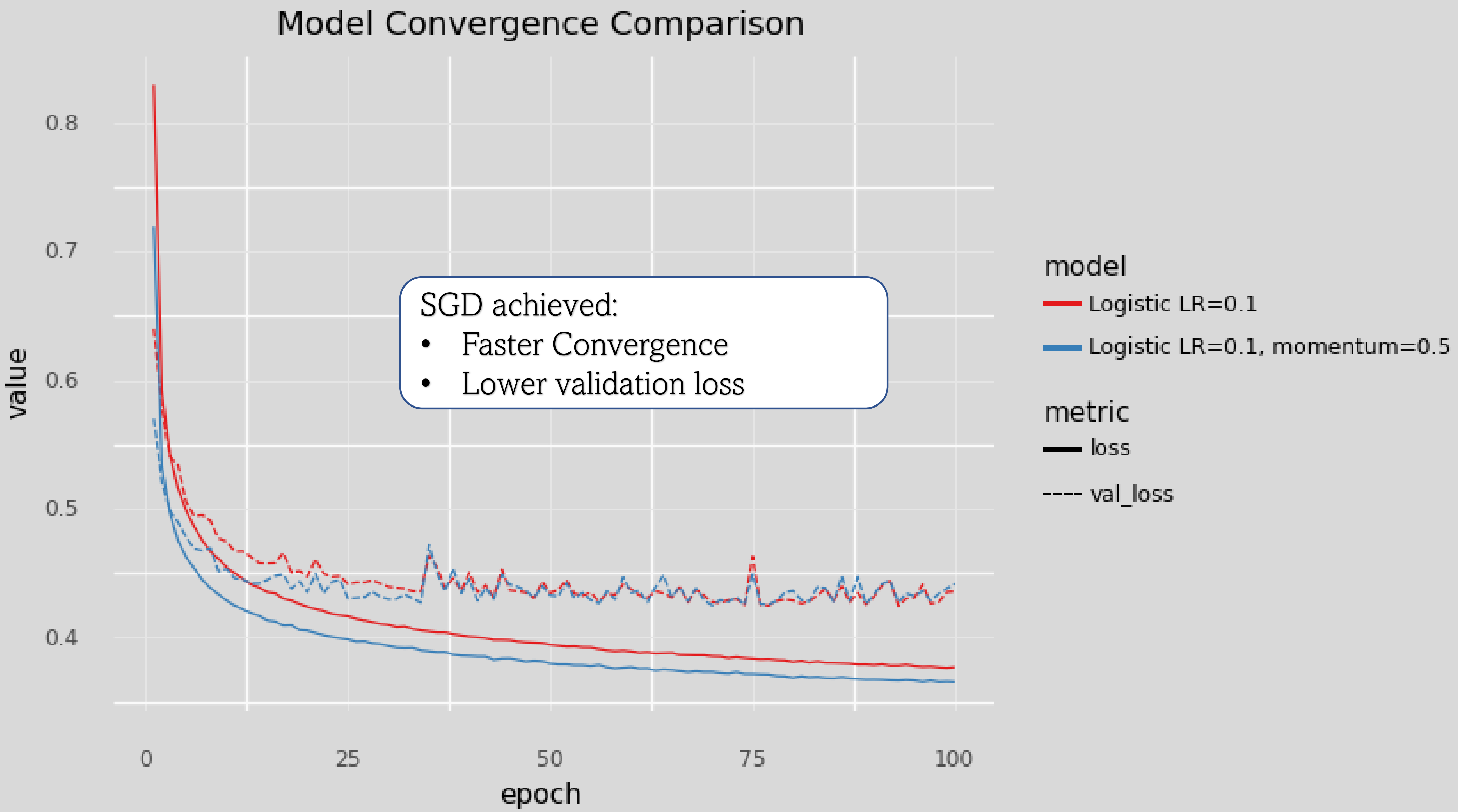


- Low learning rate
- In need of large amount of epochs

Increase lr →

- Faster convergence from epoch 50
- Not compromising the performance

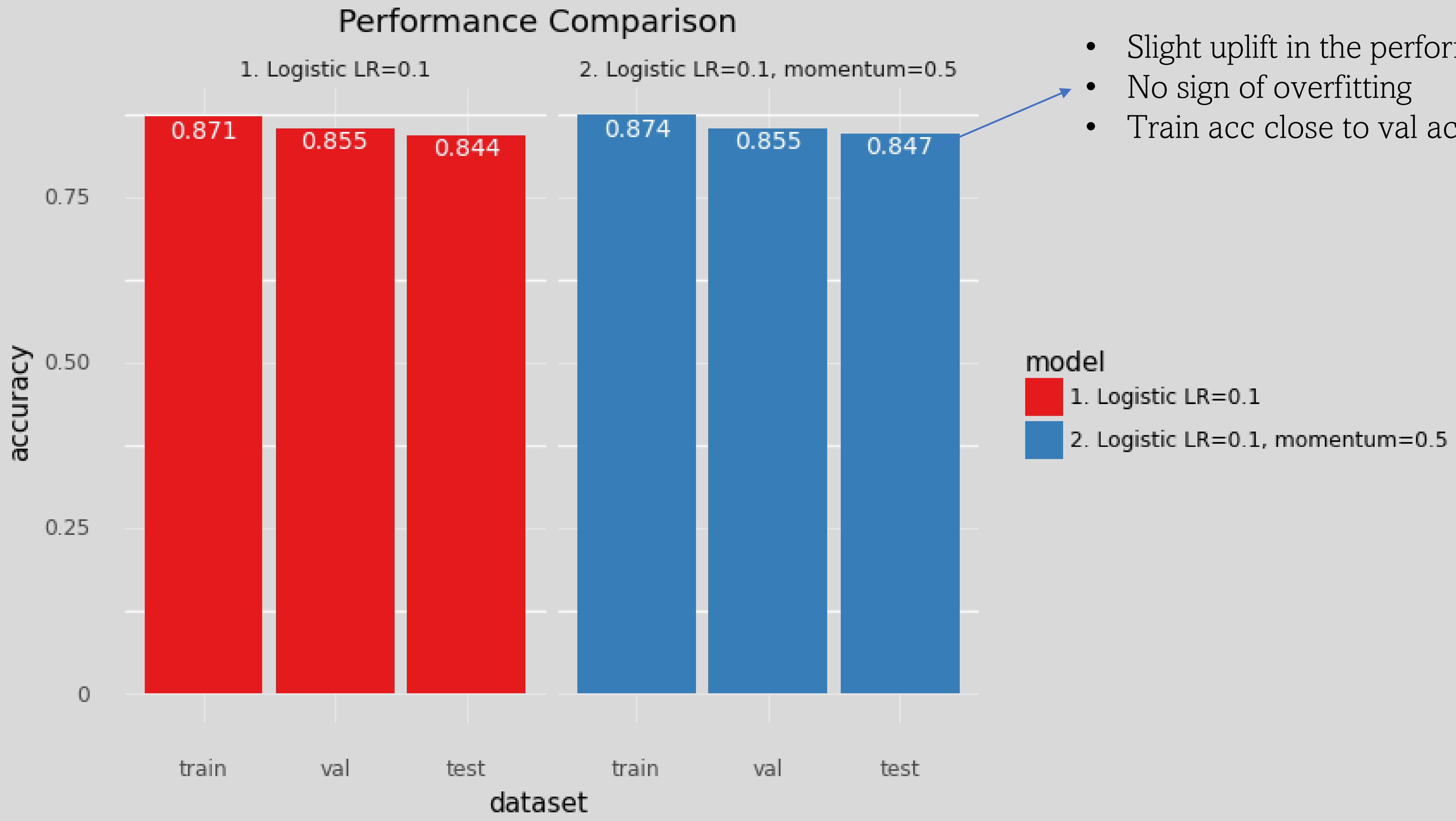
Shallow networks – SGD with Momentum



SGD achieved:

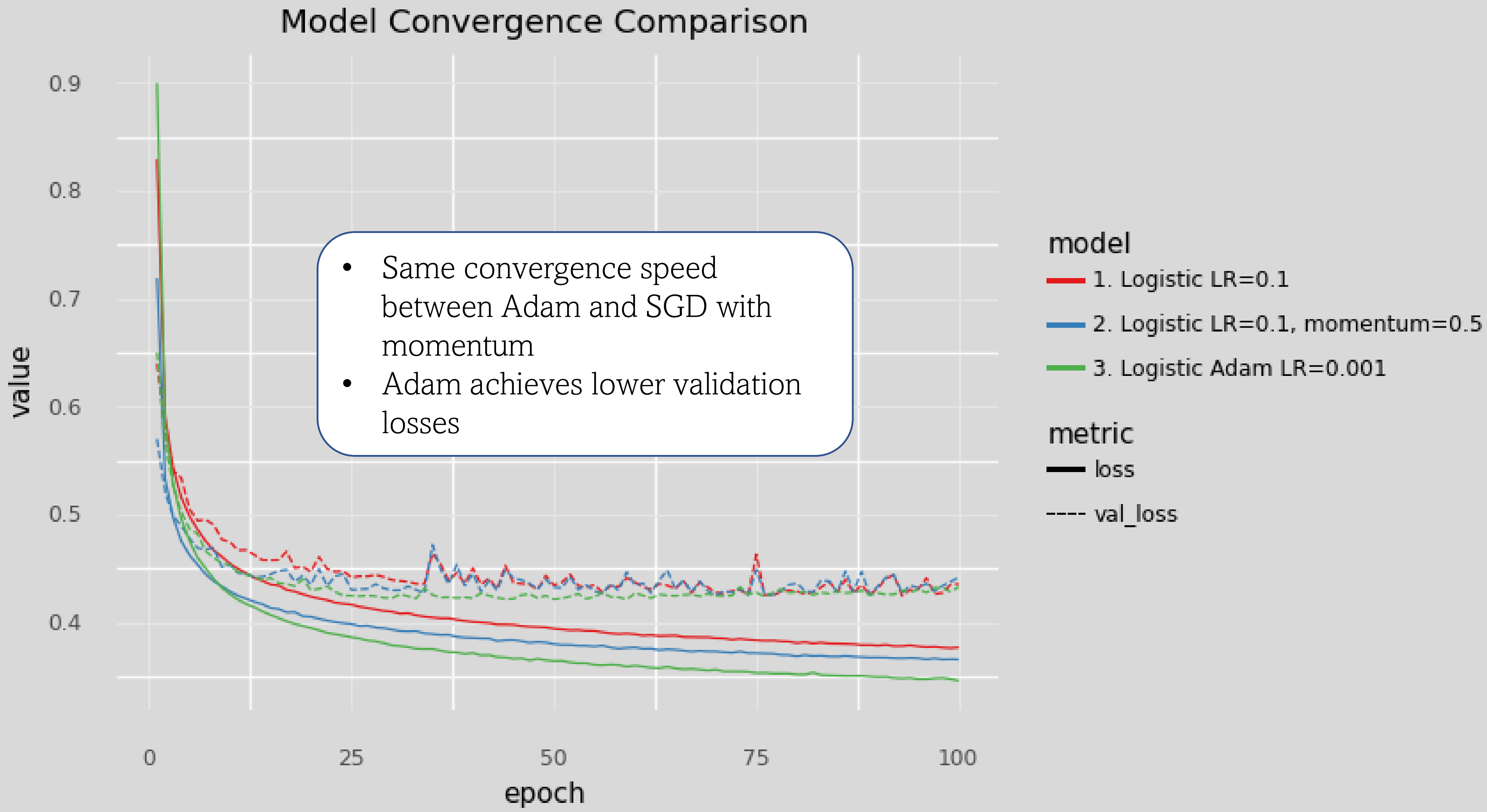
- Faster Convergence
- Lower validation loss

Shallow networks – SGD with Momentum



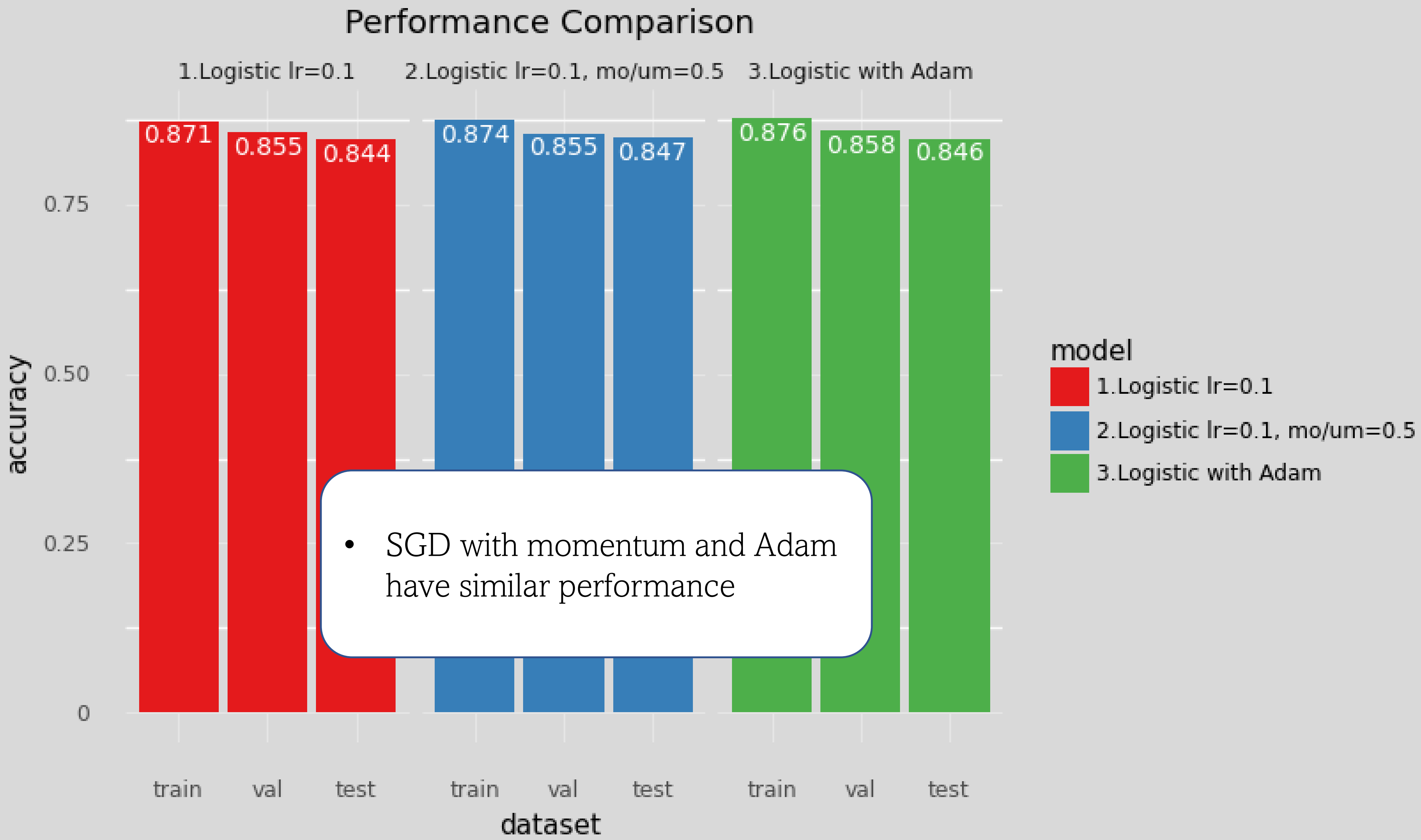
- Slight uplift in the performance
- No sign of overfitting
- Train acc close to val acc

Shallow networks – Enter Adam



- Same convergence speed between Adam and SGD with momentum
- Adam achieves lower validation losses

Shallow networks – Enter Adam



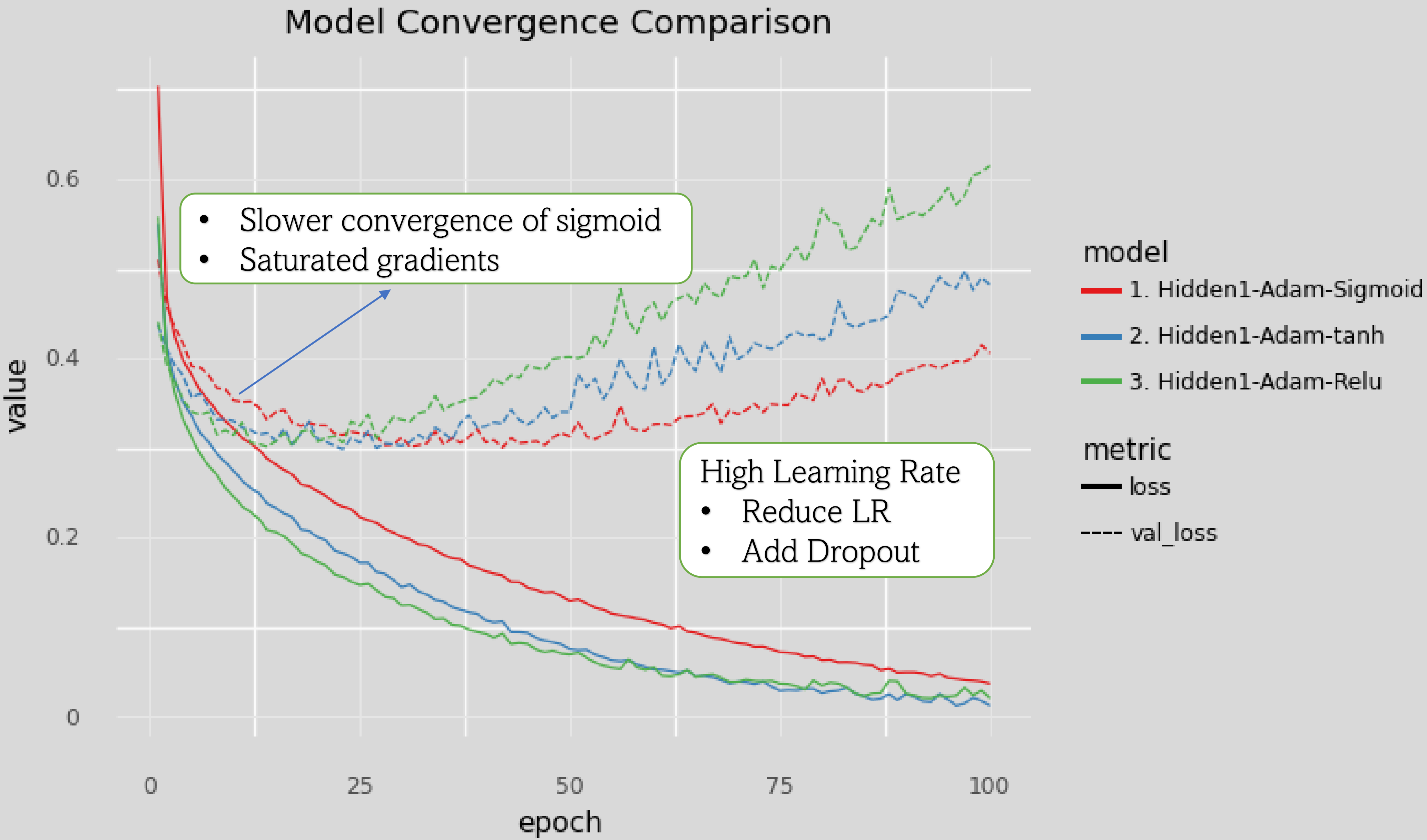
- SGD with momentum and Adam have similar performance

Shallow networks & Early Stop Patience

running_min_loss	minimum_loss_for_epochs
0.421137	42
0.422151	11
0.421286	9
0.421489	6
0.429673	3
0.422324	3
0.439099	2
0.424267	2
0.433573	2
0.459857	1

- The minimum loss achieved was **0.421137**
- In order to achieve this loss, we had to be patient for **maximum 11** epochs
- Thus **15** epochs are chosen as the Early Stopping patience parameter

1 Hidden Layer - Activations

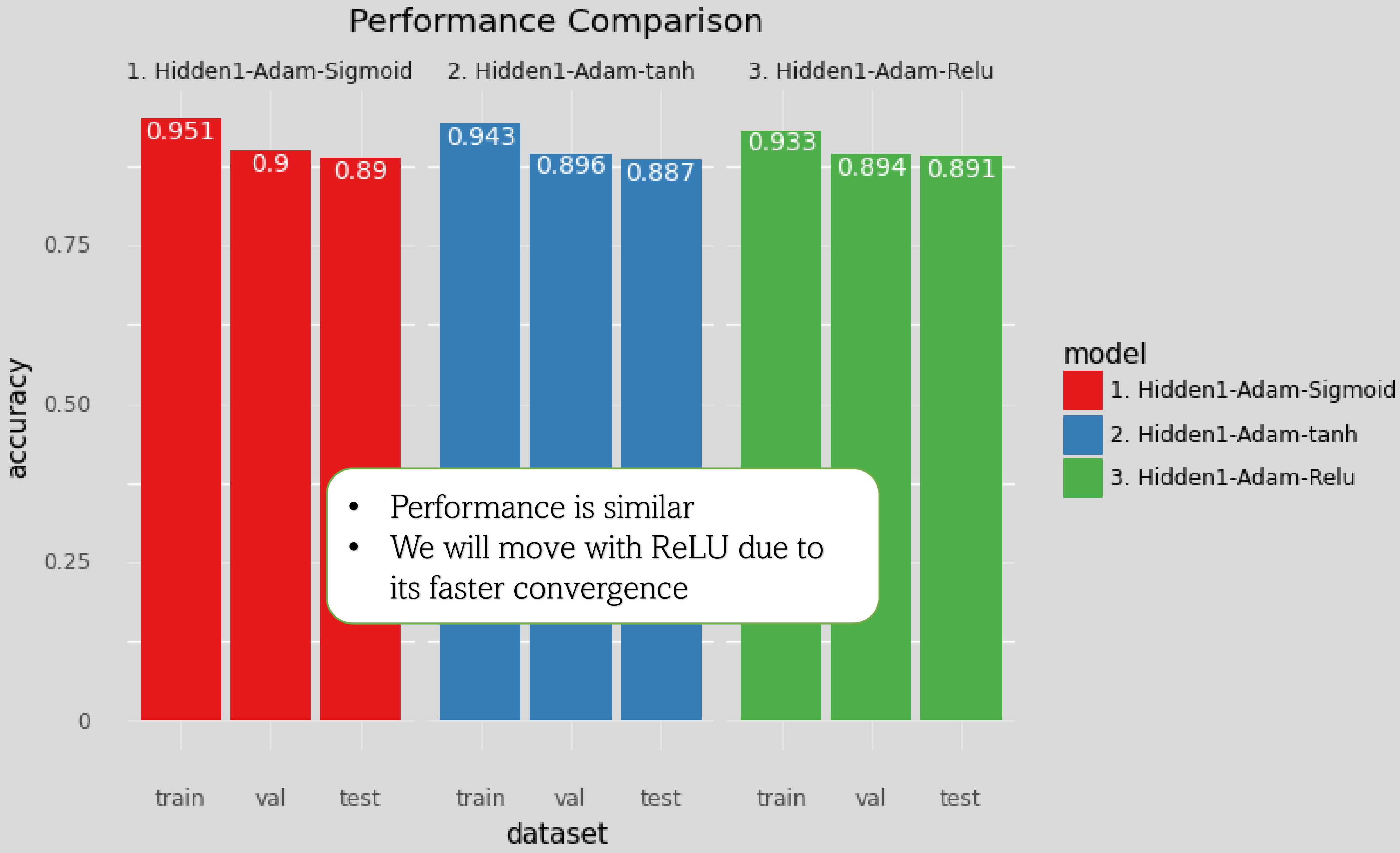


- Slower convergence of sigmoid
- Saturated gradients

High Learning Rate

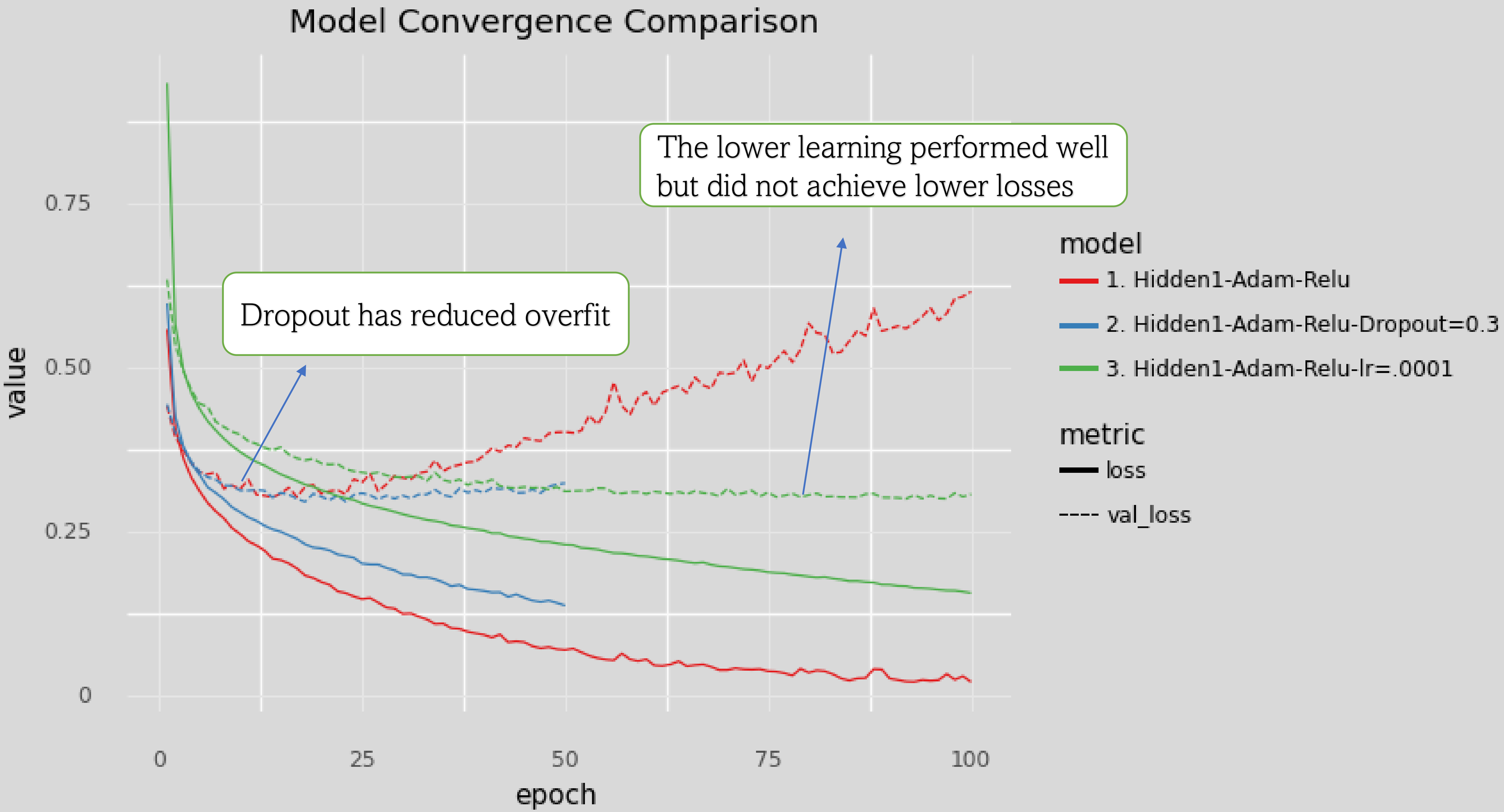
- Reduce LR
- Add Dropout

1 Hidden Layer - Activations



- Performance is similar
- We will move with ReLU due to its faster convergence

1 Hidden Layer – Dropout and lower LR

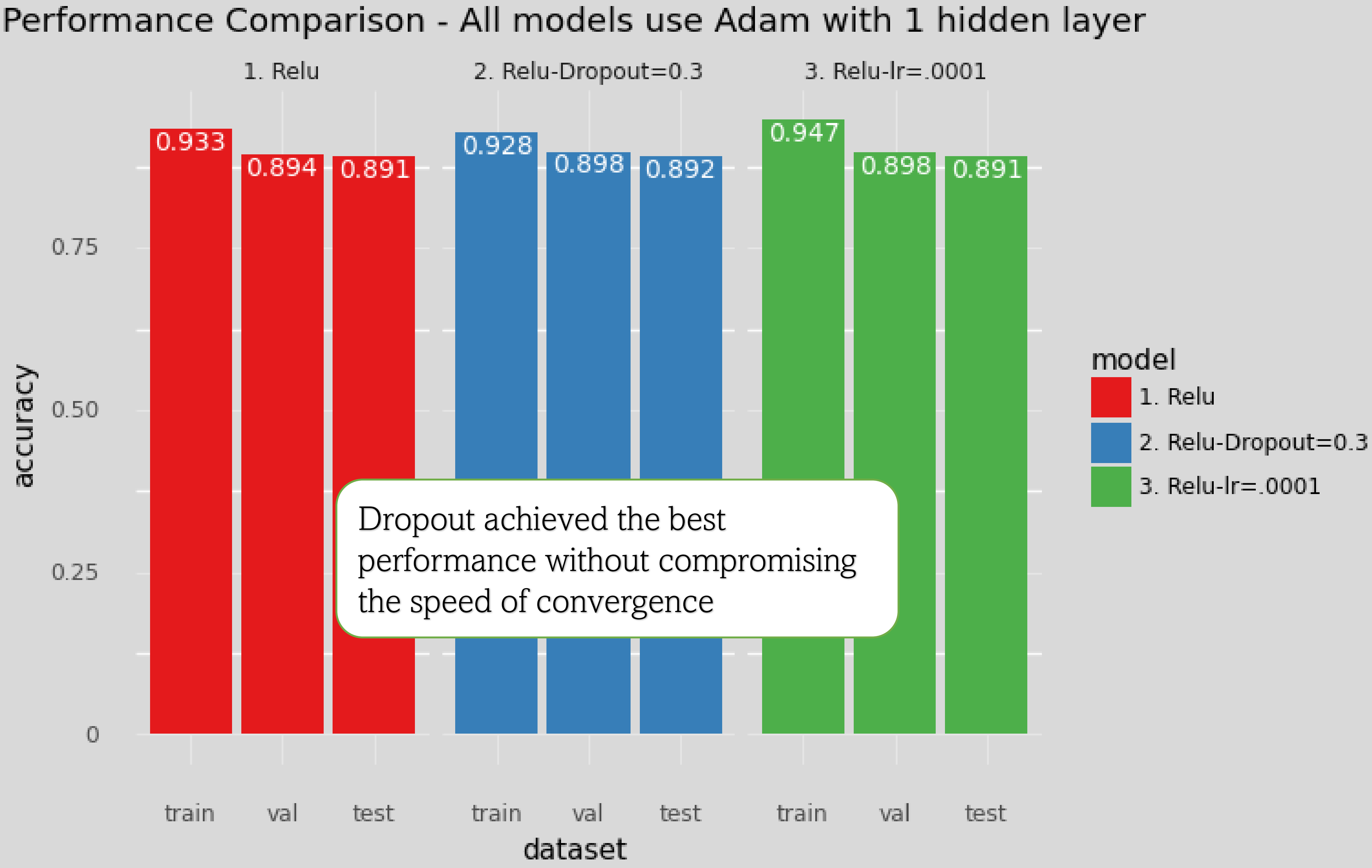


Model Convergence Comparison

Dropout has reduced overfit

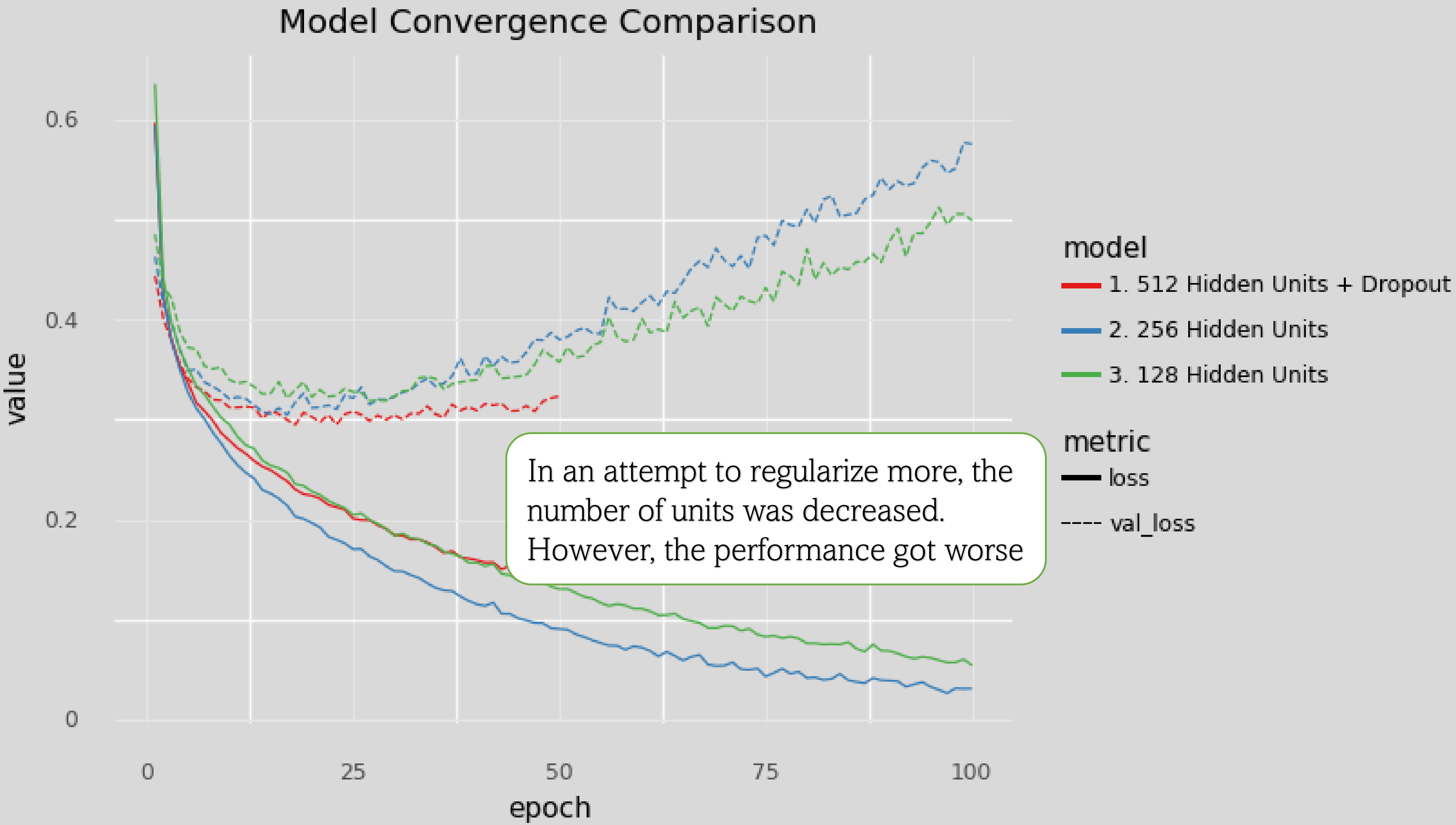
The lower learning performed well but did not achieve lower losses

1 Hidden Layer – Dropout and lower LR



Dropout achieved the best performance without compromising the speed of convergence

1 Hidden Layer – Units



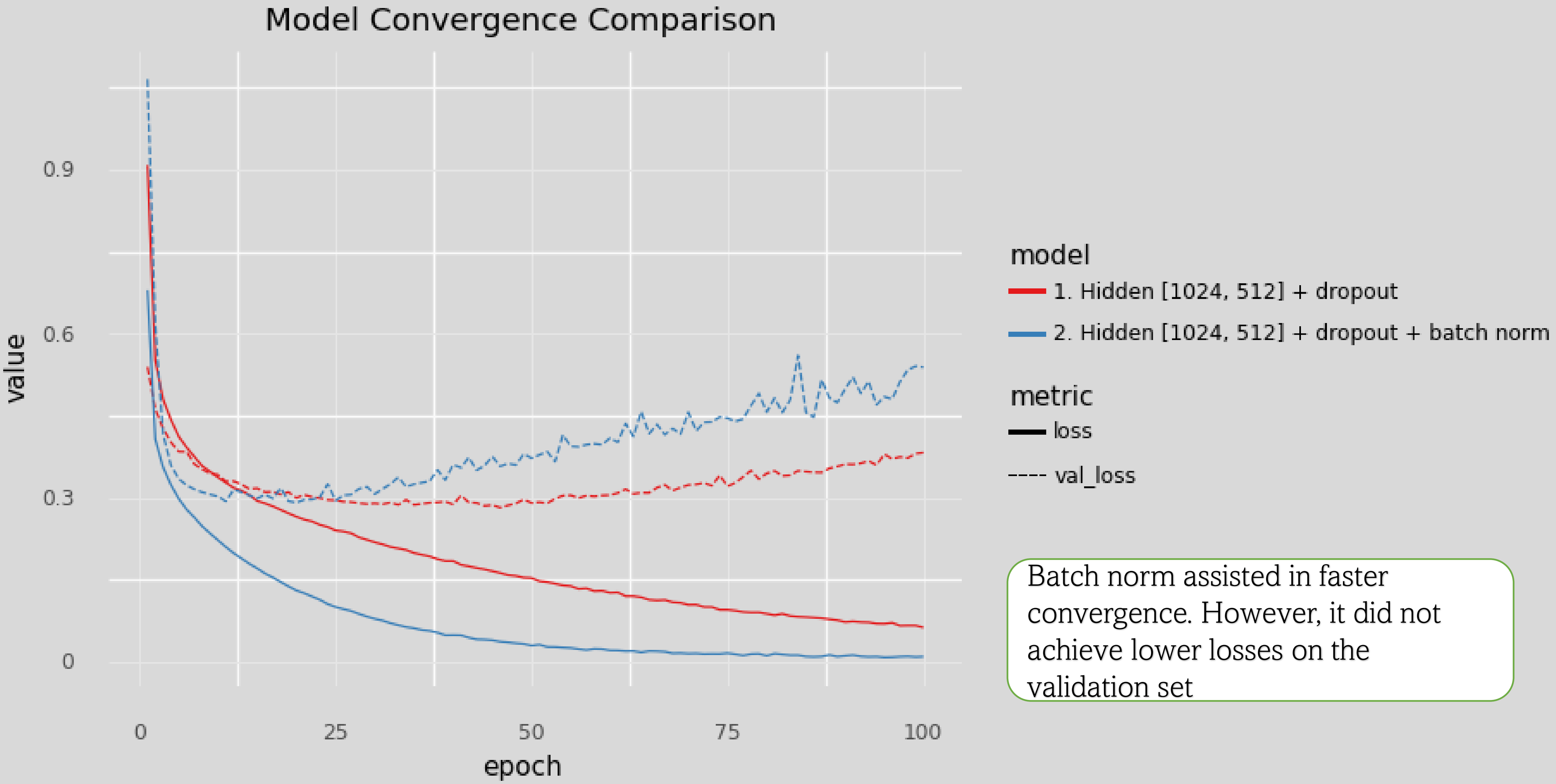
In an attempt to regularize more, the number of units was decreased. However, the performance got worse

1 Hidden Layer – Best Model

set	accuracy	precision	recall	f1
train	0.93475	0.936310	0.93475	0.934946
val	0.89925	0.900827	0.89925	0.899495
test	**0.89230**	0.894759	0.89230	**0.892482**

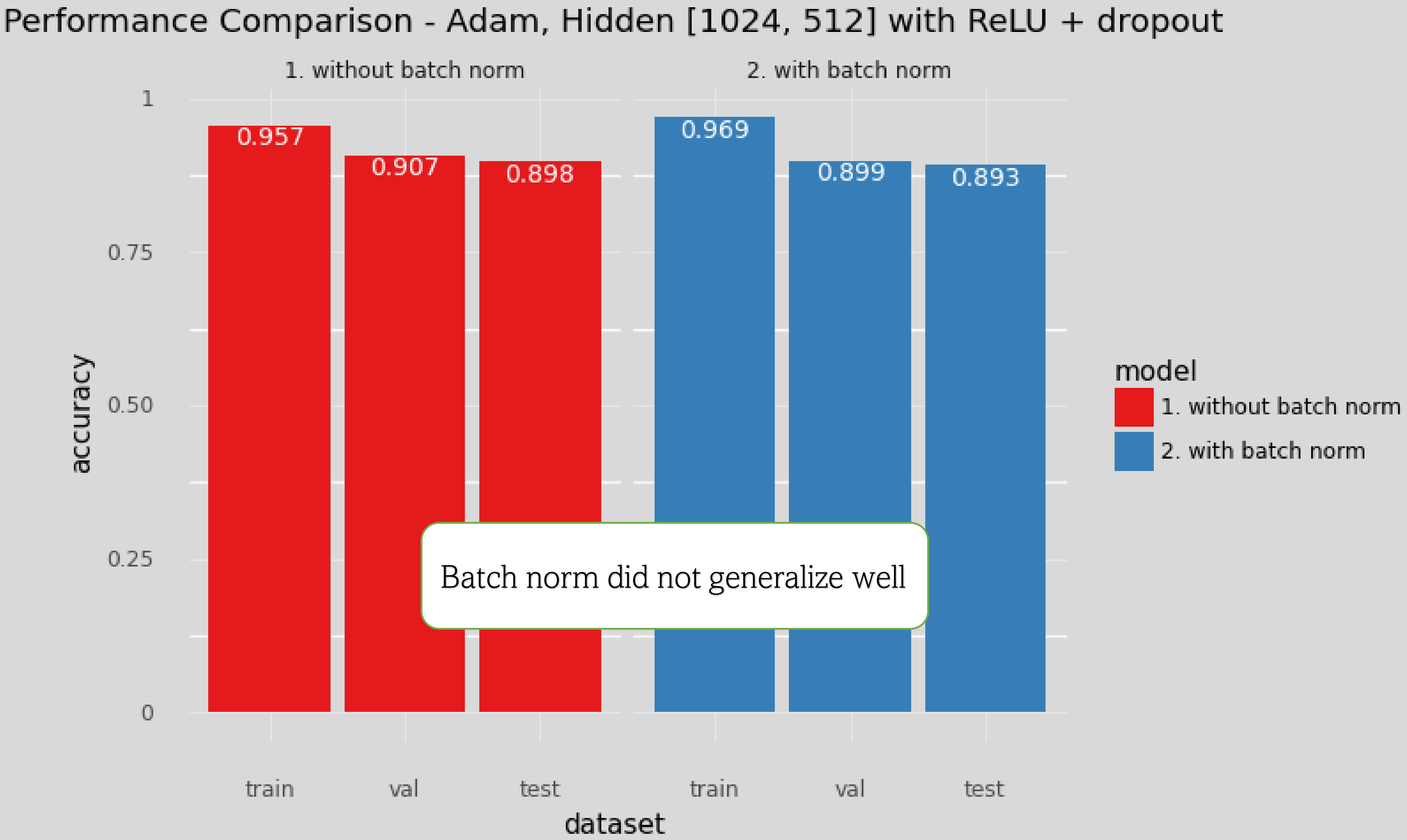
- 1 hidden layer
- 512 Units
- ReLU activation
- 0.001 Learning Rate
- Adam Optimizer

2 Hidden Layers – Batch norm



Batch norm assisted in faster convergence. However, it did not achieve lower losses on the validation set

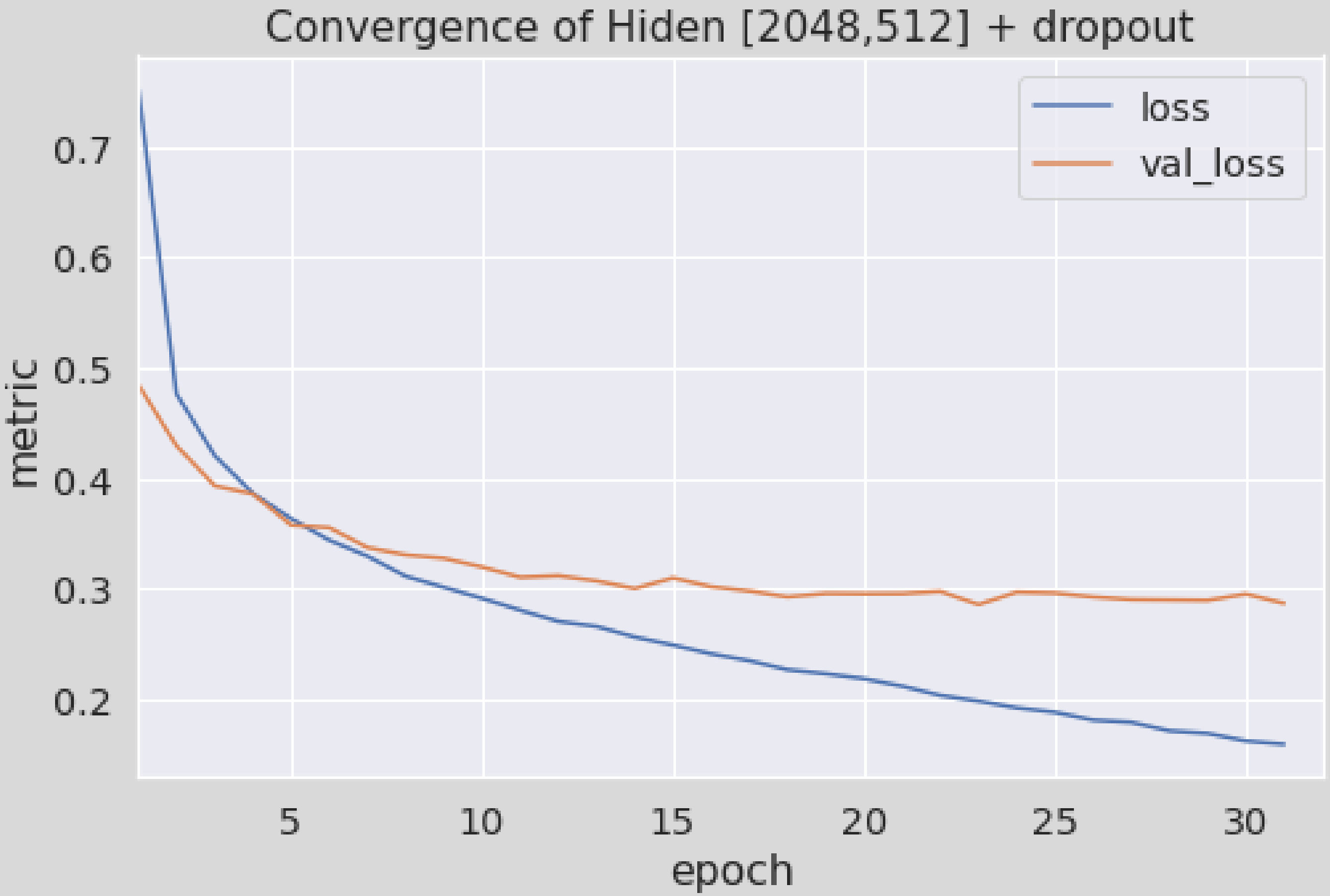
2 Hidden Layers – Batch norm



Performance Comparison - Adam, Hidden [1024, 512] with ReLU + dropout

Batch norm did not generalize well

Hypertuning an MLP



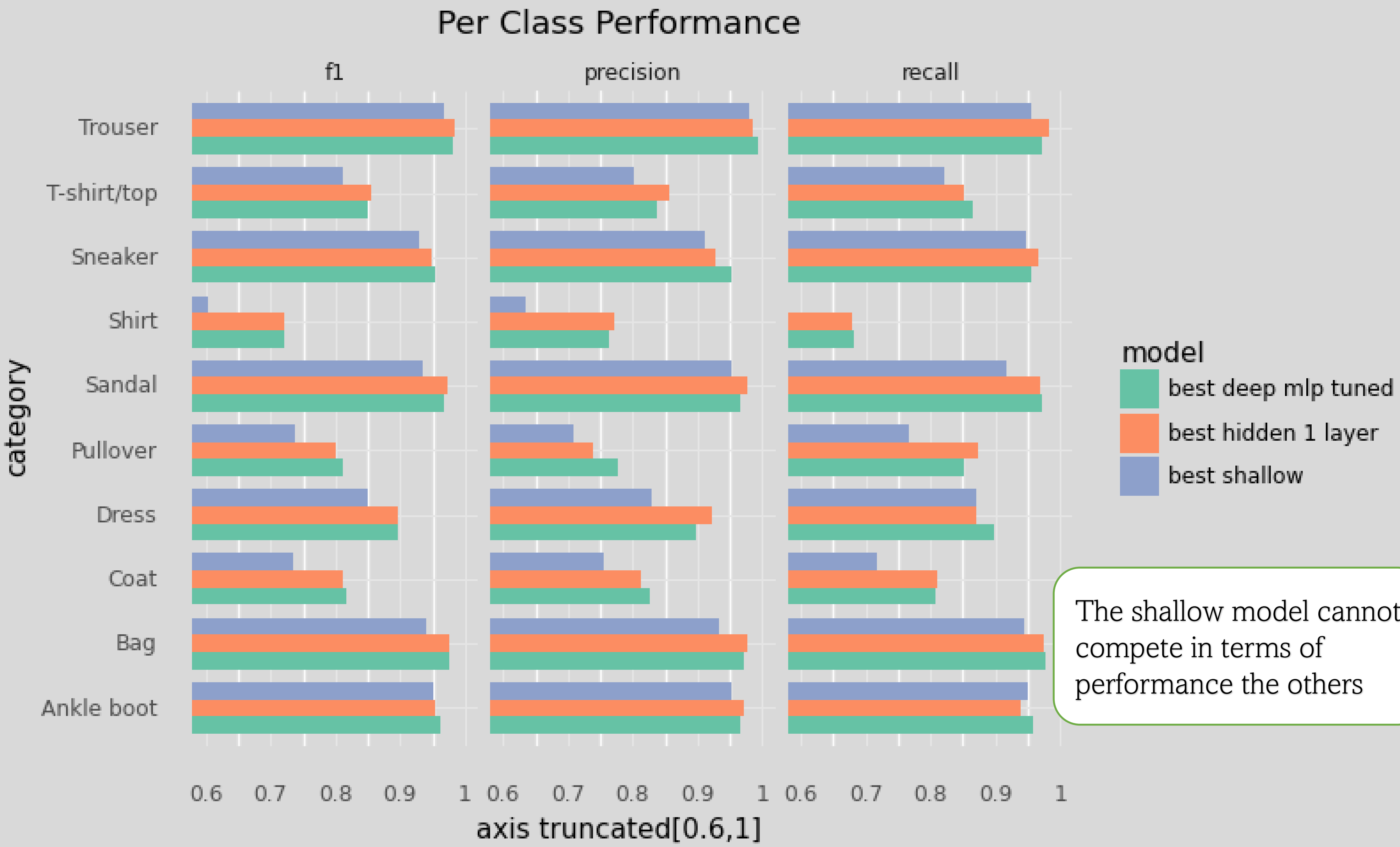
- 2 Hidden layers with
- 2048 and 512 units respectively
- ReLU activation
- Dropouts 0.2 and 0.4 respectively
- Adam with 0.0001 learning rate

set	accuracy	precision	recall	f1
train	0.940625	0.941899	0.940625	0.940899
val	0.901500	0.902927	0.901500	0.901947
test	**0.894400**	0.896246	0.894400	**0.894873**

Per class Performance

Looking at the f1 scores:

- In some classes the tuned MLP has better performance
- In other classes the model with 1 hidden layers has better performance
- **Combine them**



The shallow model cannot compete in terms of performance the others

MLP Ensemble

$$\text{Argmax}\left(\ \text{Softmax}\left(\ \text{Probabilities}_{mlp1} + \text{Probabilities}_{mlp2}\ \right)\ \right)$$

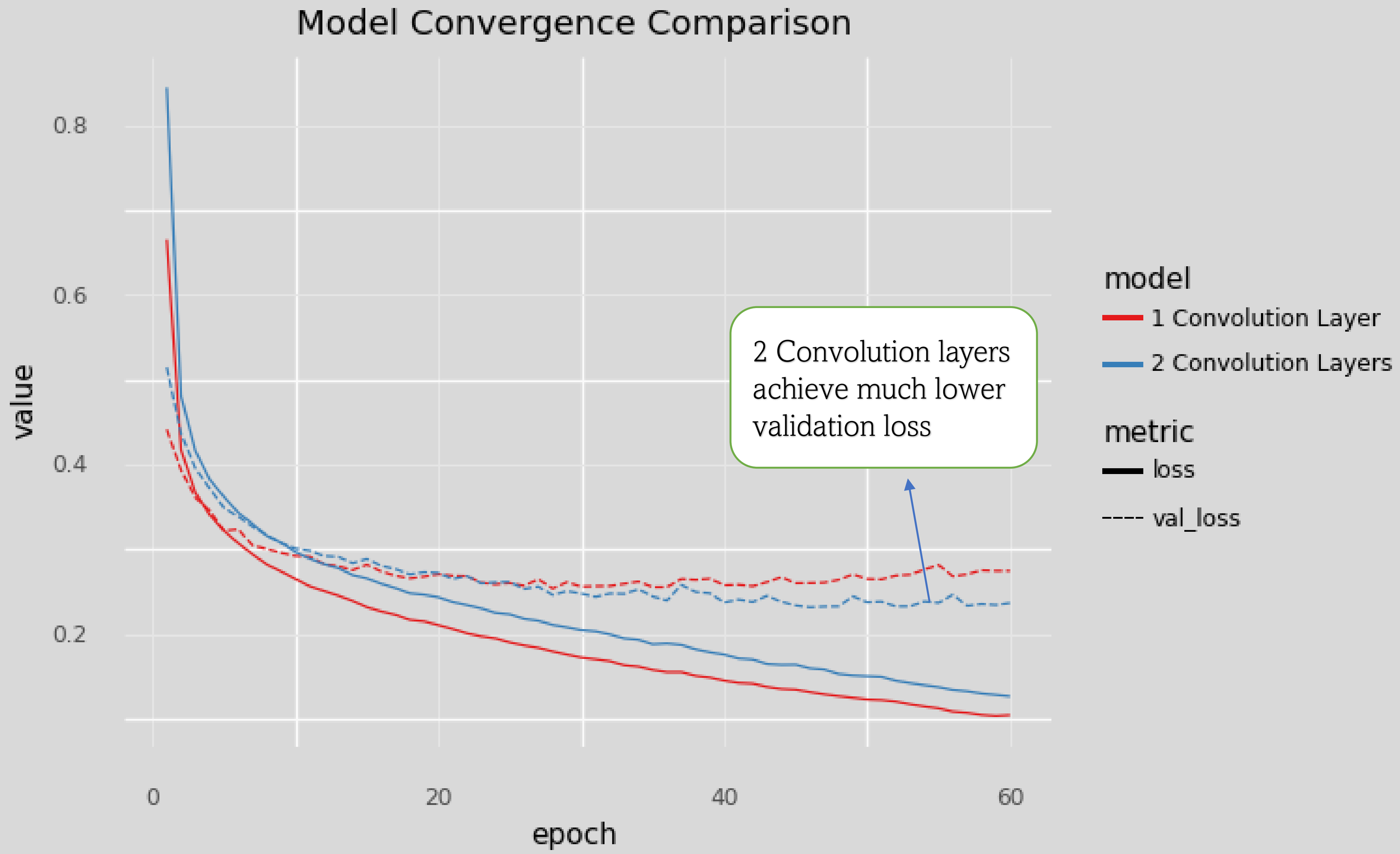
Where:

- $\text{Probabilities}_{mlp1}$ is the SoftMax output of the tuned mlp
- $\text{Probabilities}_{mlp2}$ is the SoftMax output of the best model with 1 Hidden layer

The ensemble outperforms the separate performance of its components

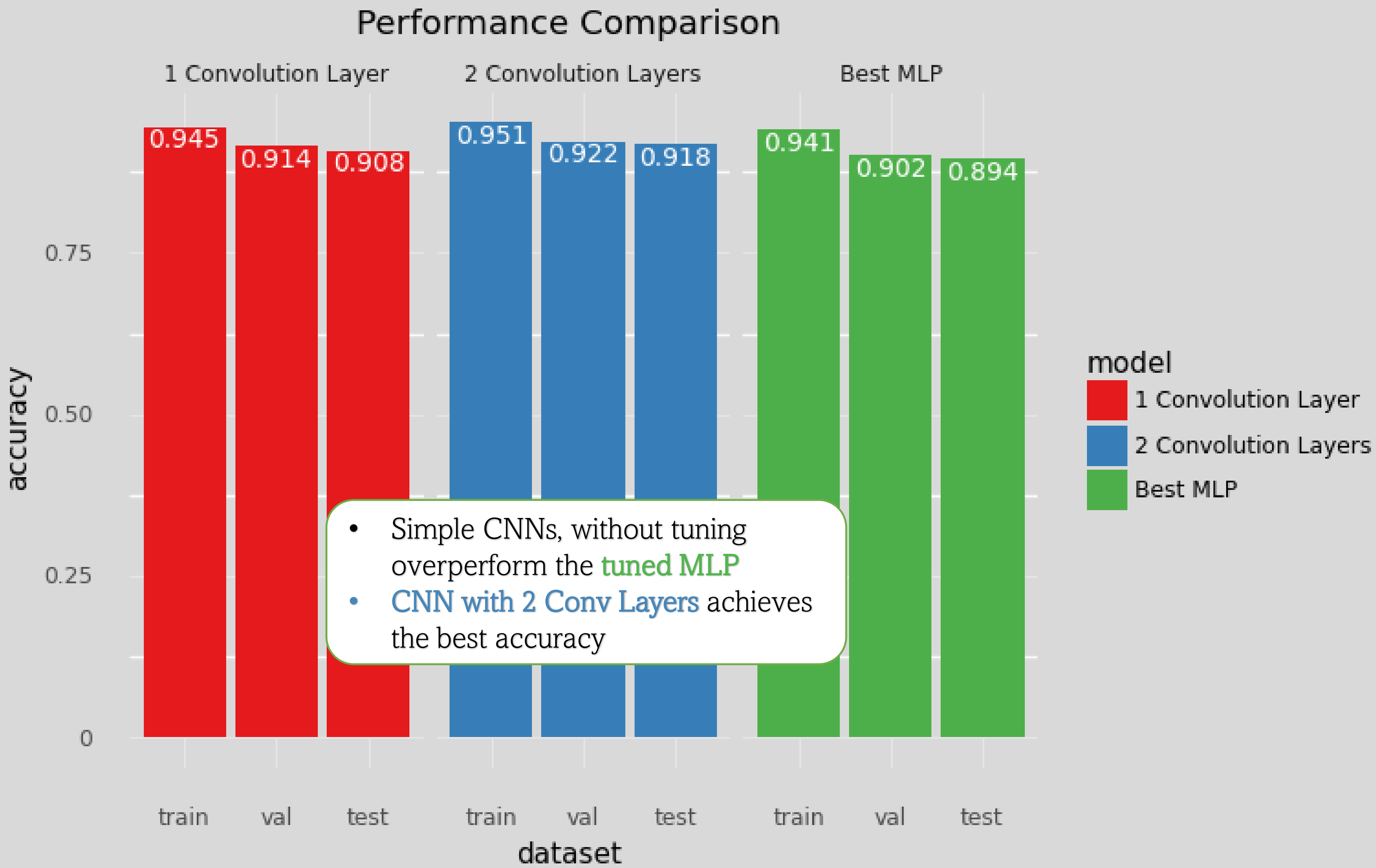
model	score
Hidden [512] + dropout	0.8923
Hidden [2048, 512] + dropout	0.8944
Ensemble	0.8960

CNNs – Convolution Layers



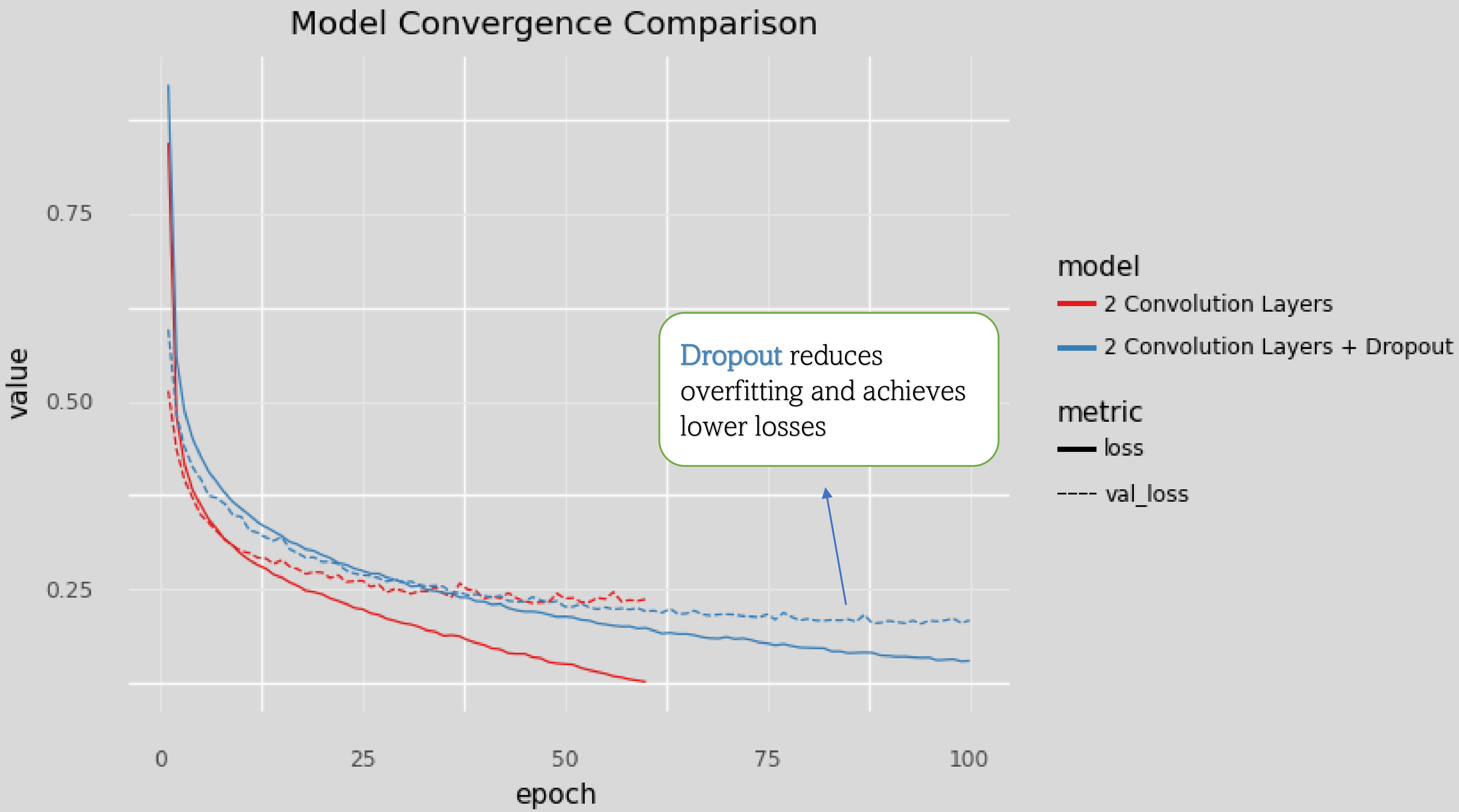
2 Convolution layers achieve much lower validation loss

CNNs vs Tuned MLP



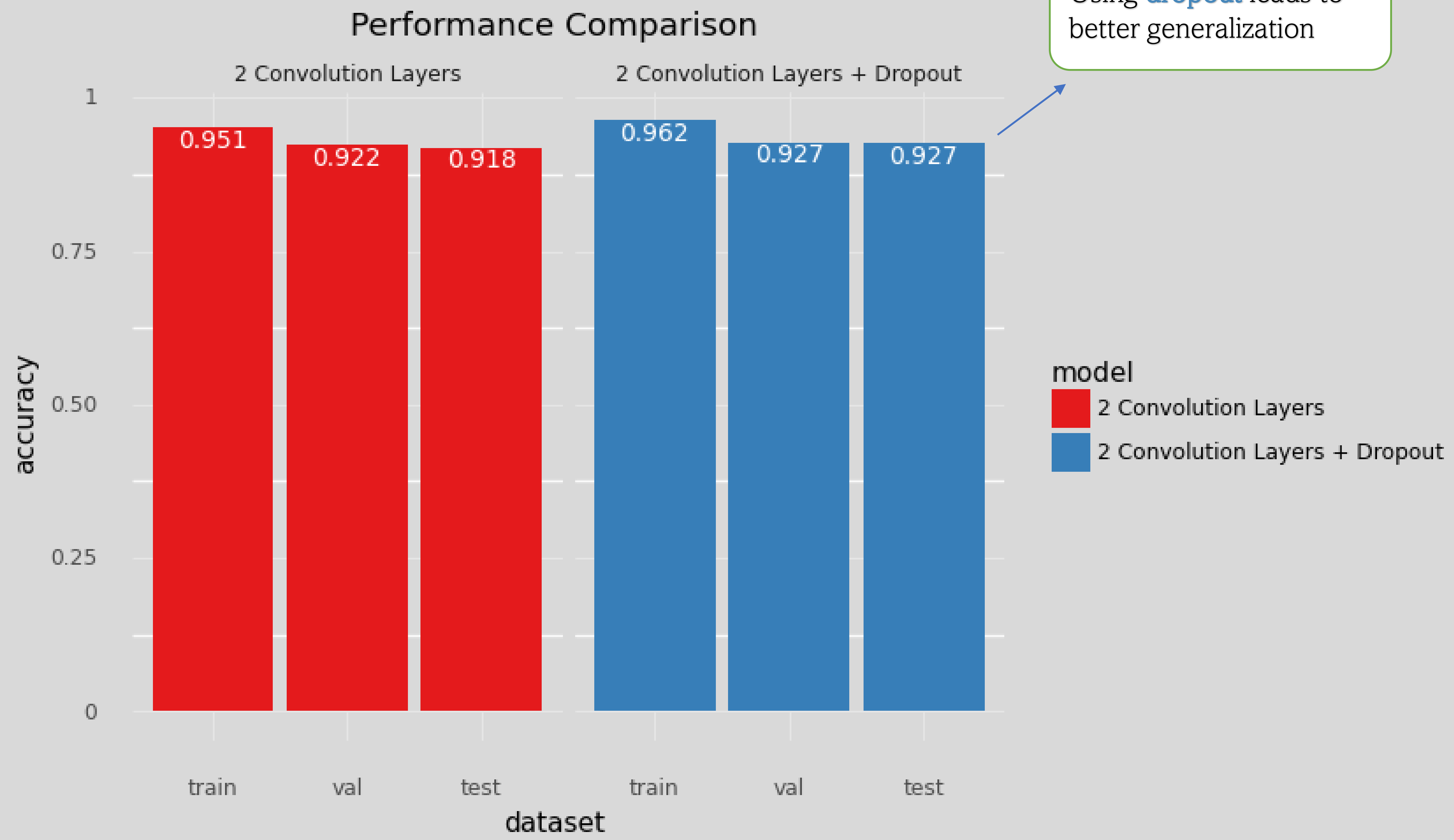
- Simple CNNs, without tuning overperform the tuned MLP
- CNN with 2 Conv Layers achieves the best accuracy

CNNs – Dropout



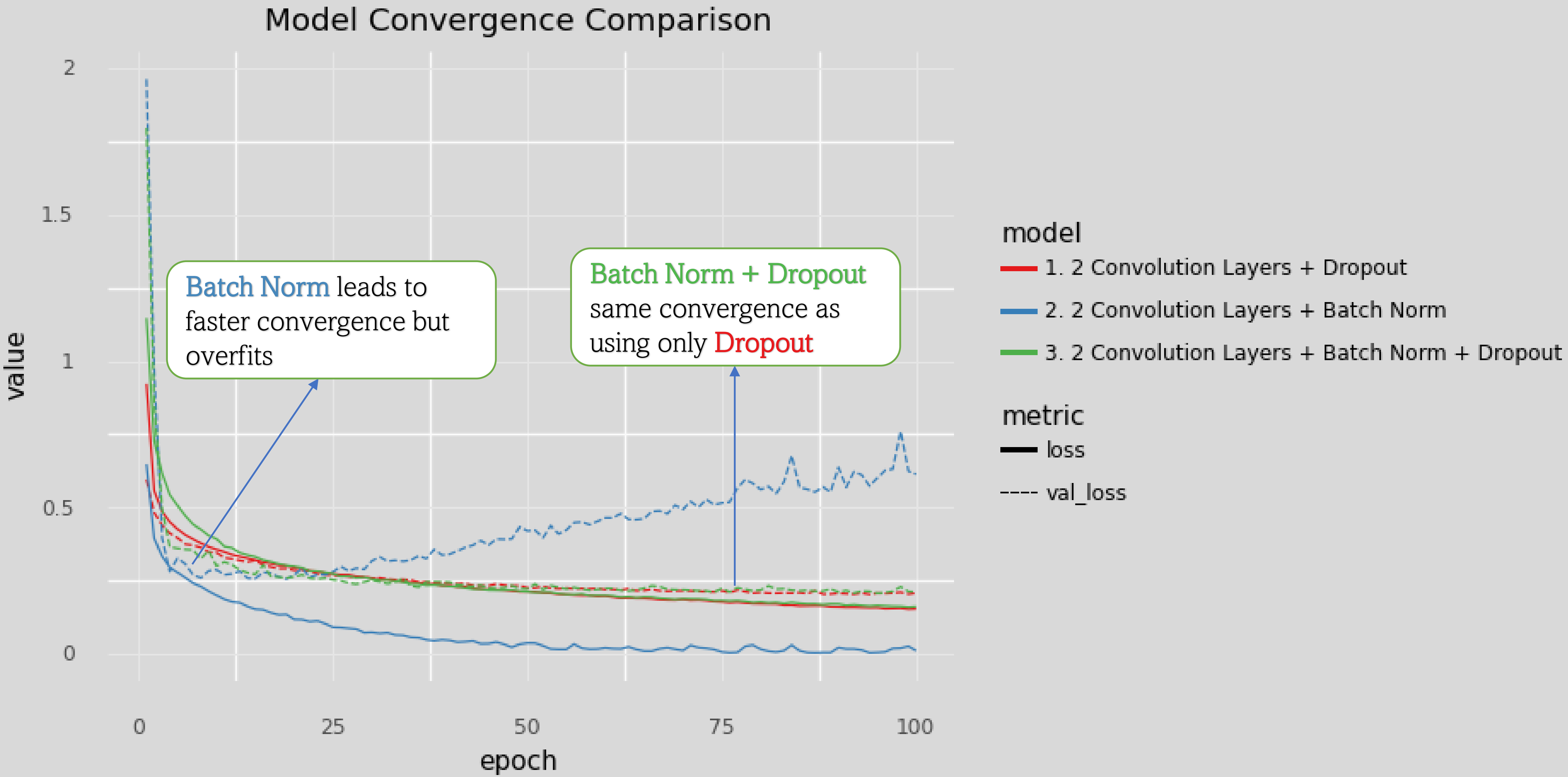
Dropout reduces overfitting and achieves lower losses

CNNs – Dropout



Using dropout leads to better generalization

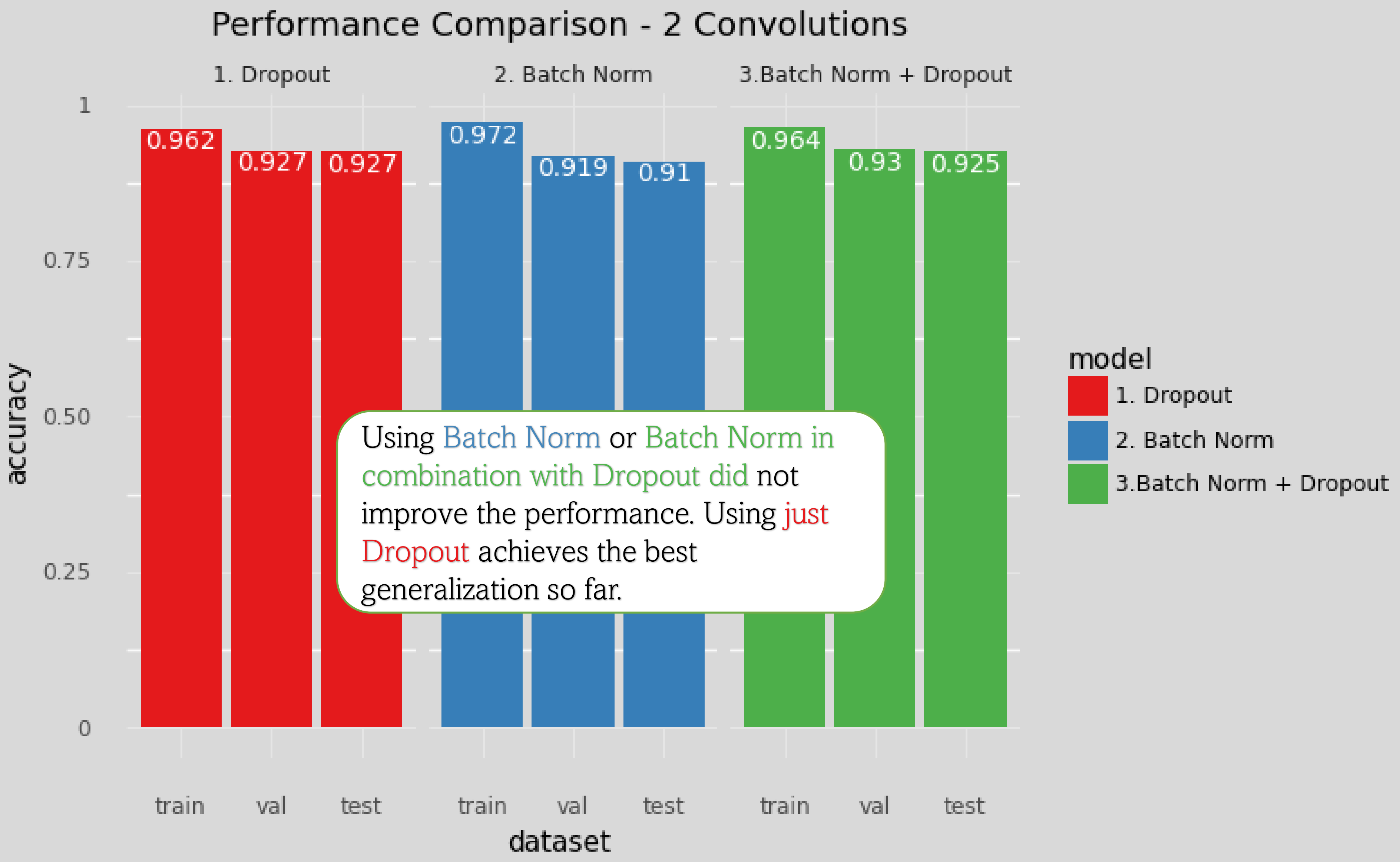
CNNs – Batch Normalization



Batch Norm leads to faster convergence but overfits

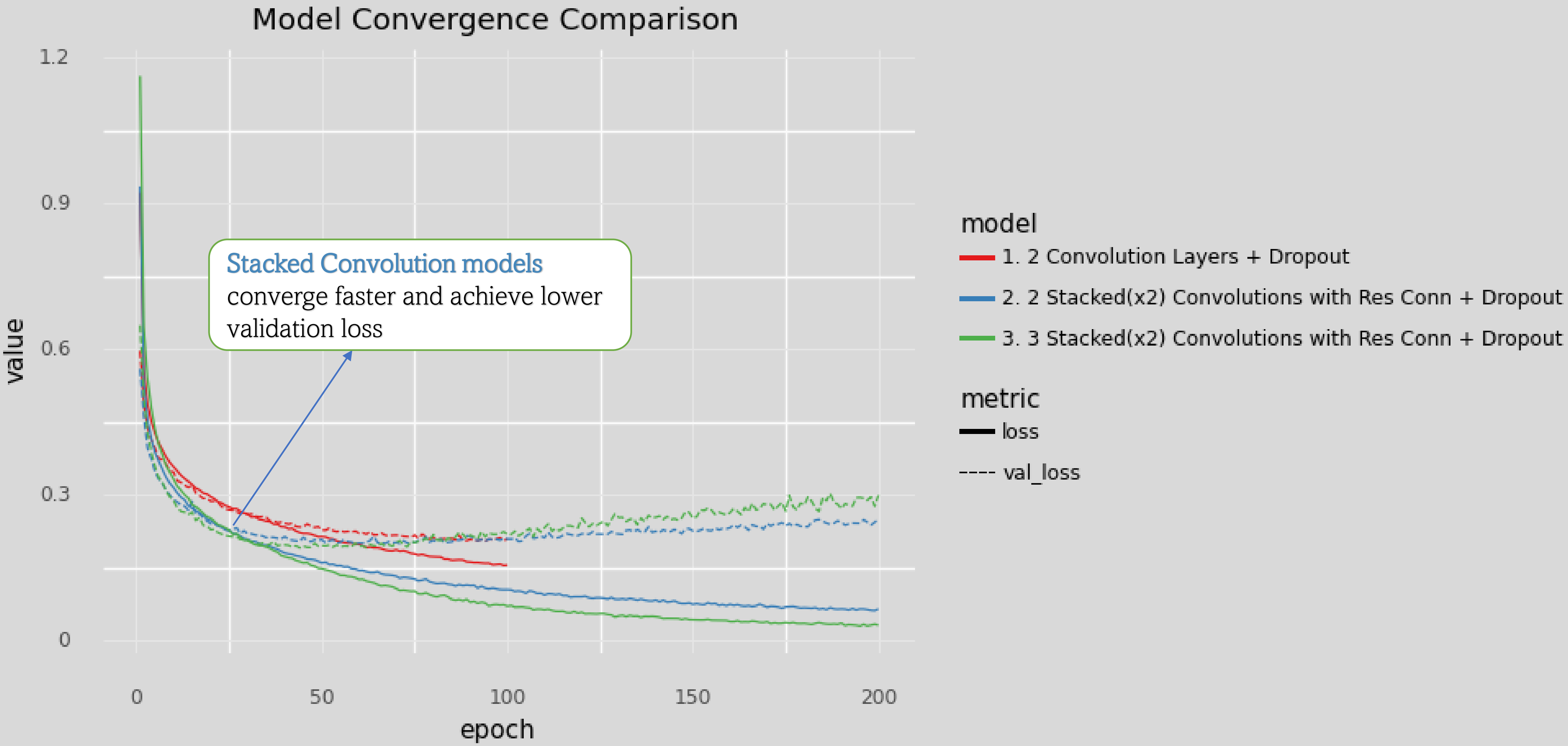
Batch Norm + Dropout same convergence as using only Dropout

CNNs – Batch Normalization



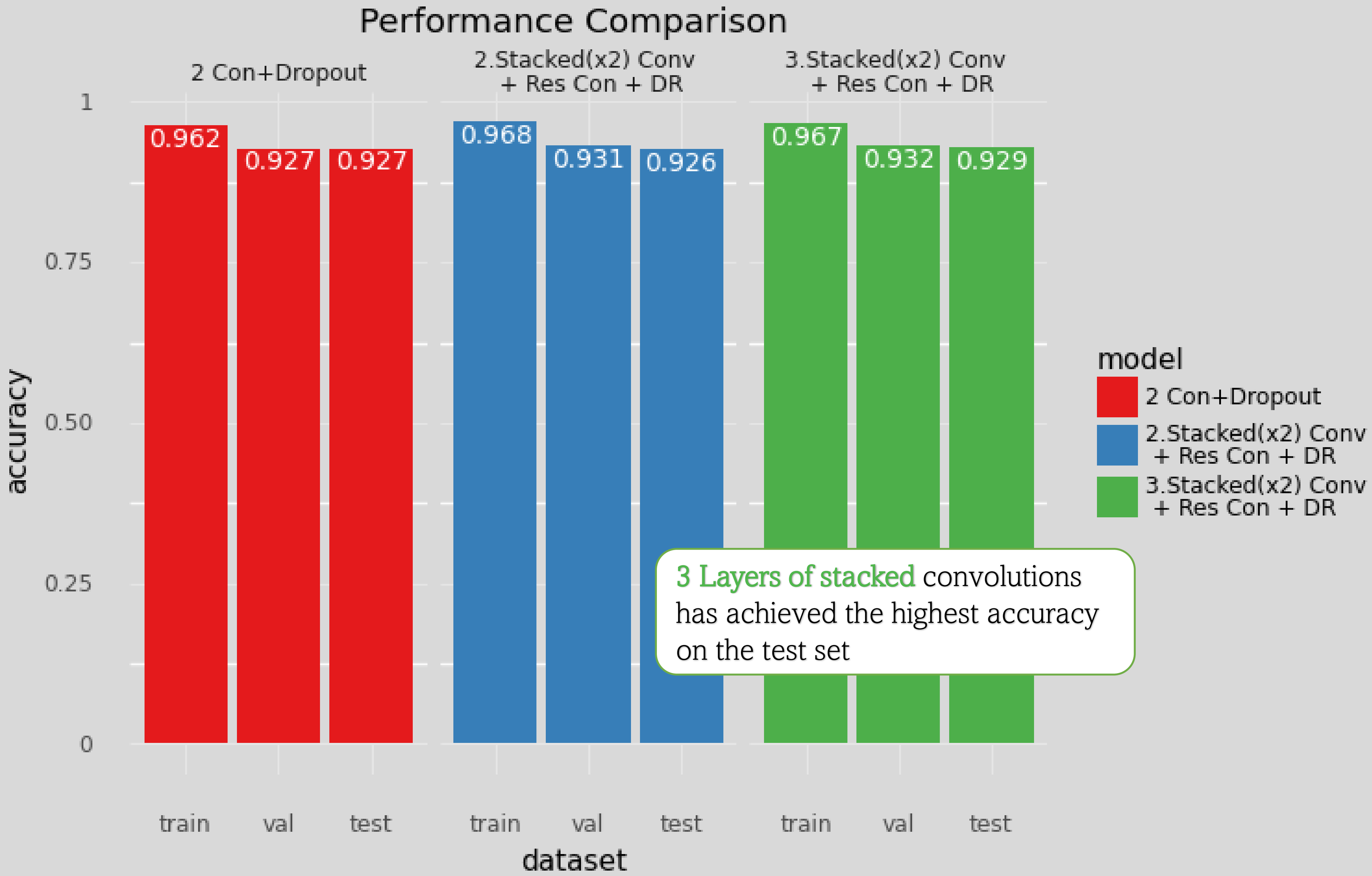
Using Batch Norm or Batch Norm in combination with Dropout did not improve the performance. Using just Dropout achieves the best generalization so far.

CNNs – Stacked Convolutions with Residuals



Stacked Convolution models converge faster and achieve lower validation loss

CNNs – Stacked Convolutions with Residuals



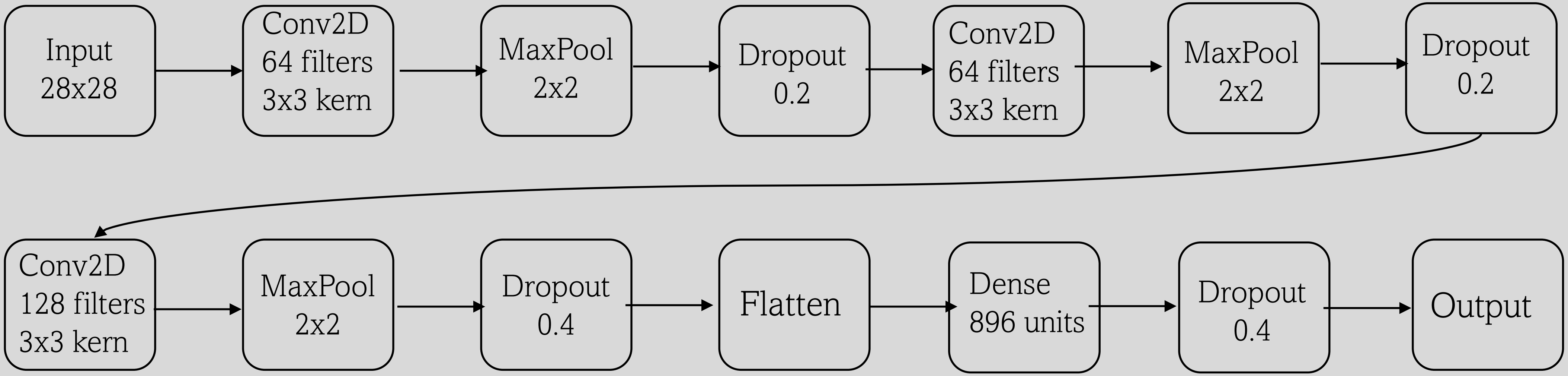
3 Layers of stacked convolutions has achieved the highest accuracy on the test set

Hypertuning CNNs – Early Stop Patience

running_min	minimum_loss_for_period
0.203920	7
0.204610	6
0.208205	4
0.215054	4
0.233148	4
0.209780	3
0.216725	3
0.314627	2
0.260298	2
0.258610	2

- The minimum loss achieved was **0.2039**
- In order to achieve this loss, we had to be patient for **maximum 6** epochs
- Thus **10** epochs are chosen as the Early Stopping patience parameter

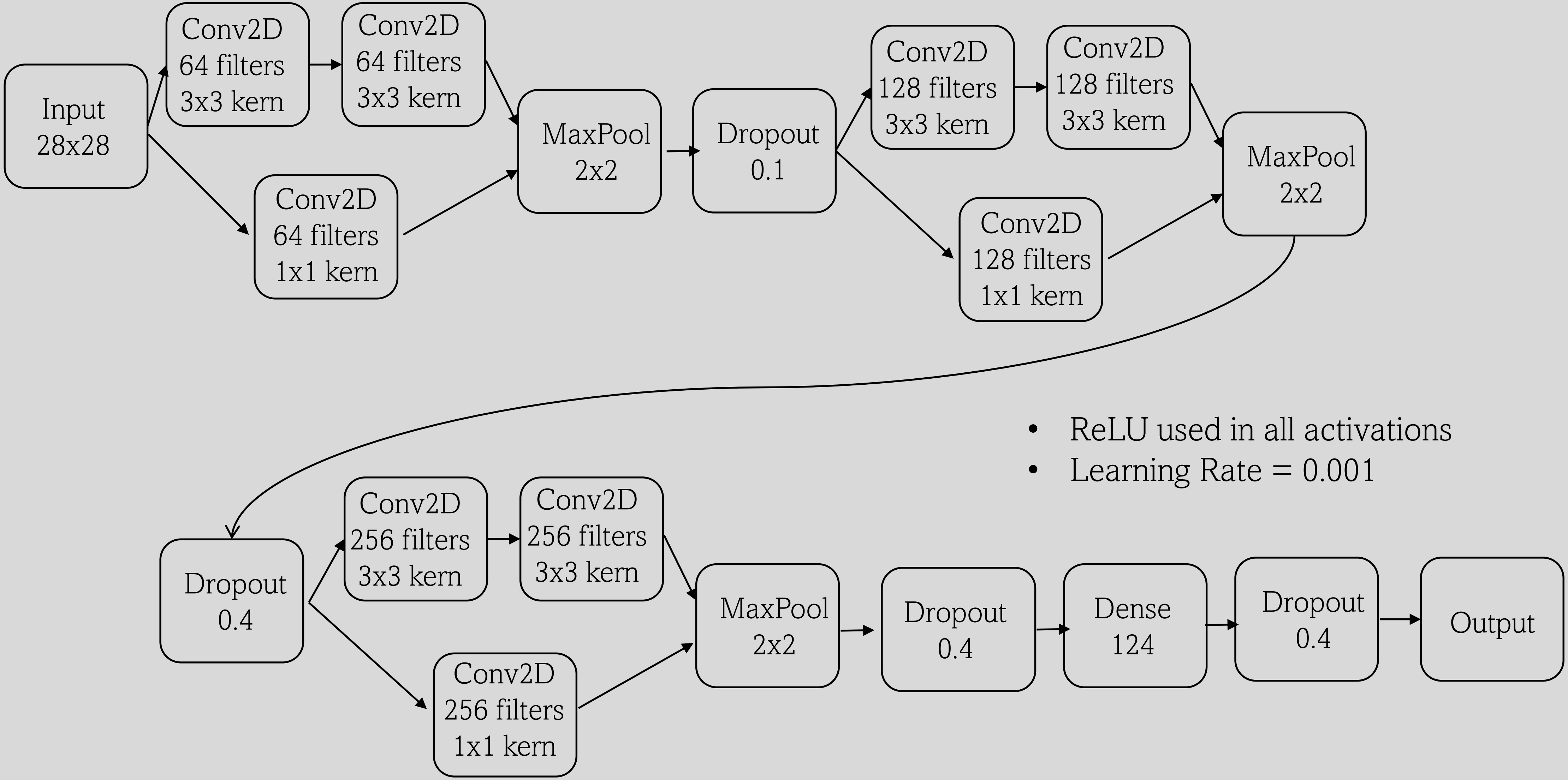
Hypertuning a CNN



- ReLU used in all activations
- Learning Rate = 0.001

set	accuracy	precision	recall	f1
train	0.969708	0.970038	0.969708	0.969801
val	0.933167	0.933886	0.933167	0.933414
test	**0.926800**	0.927751	0.926800	**0.927089**

Hypertuning a Stacked CNN - Architecture

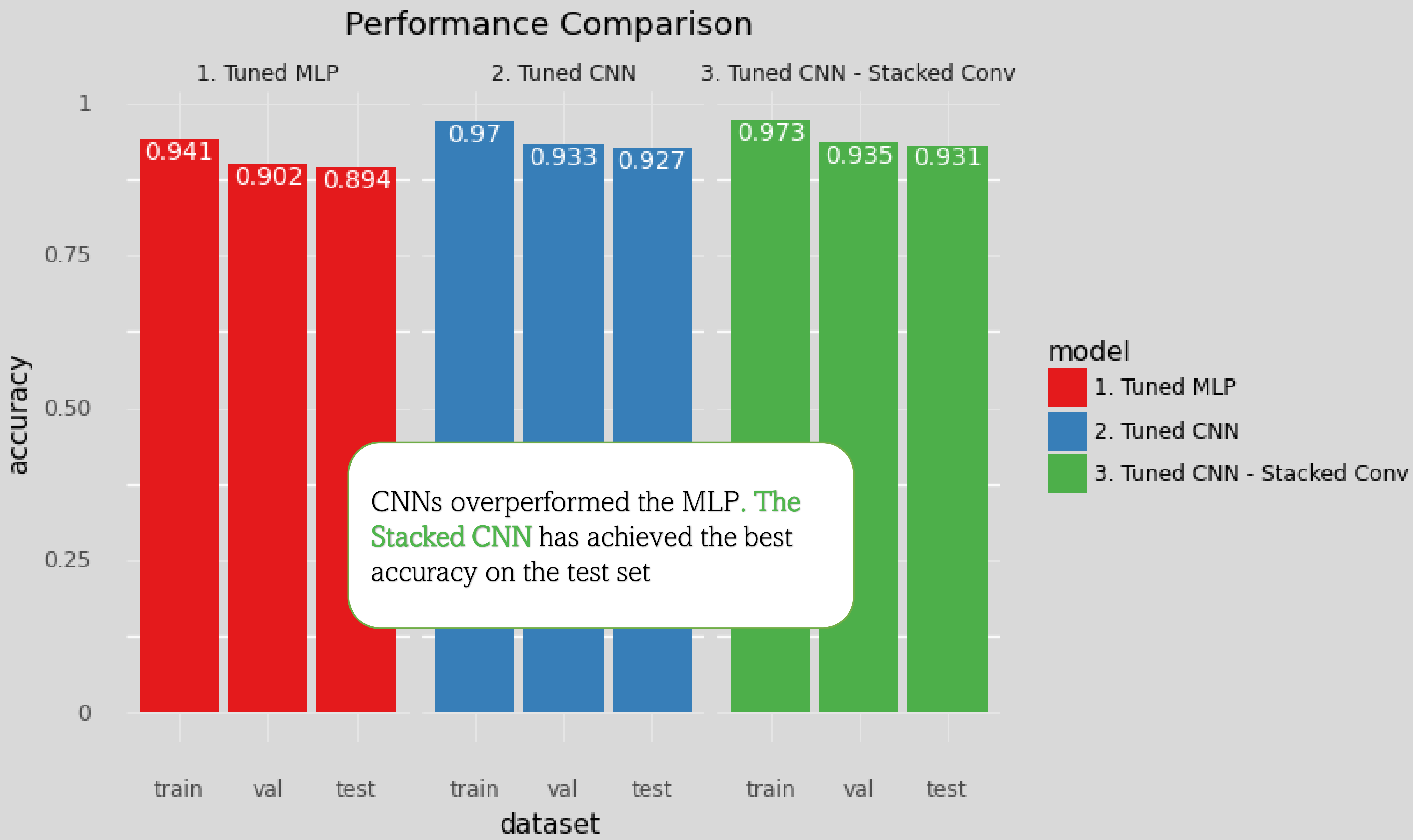


- ReLU used in all activations
- Learning Rate = 0.001

Hypertuning a Stacked CNN - Performance

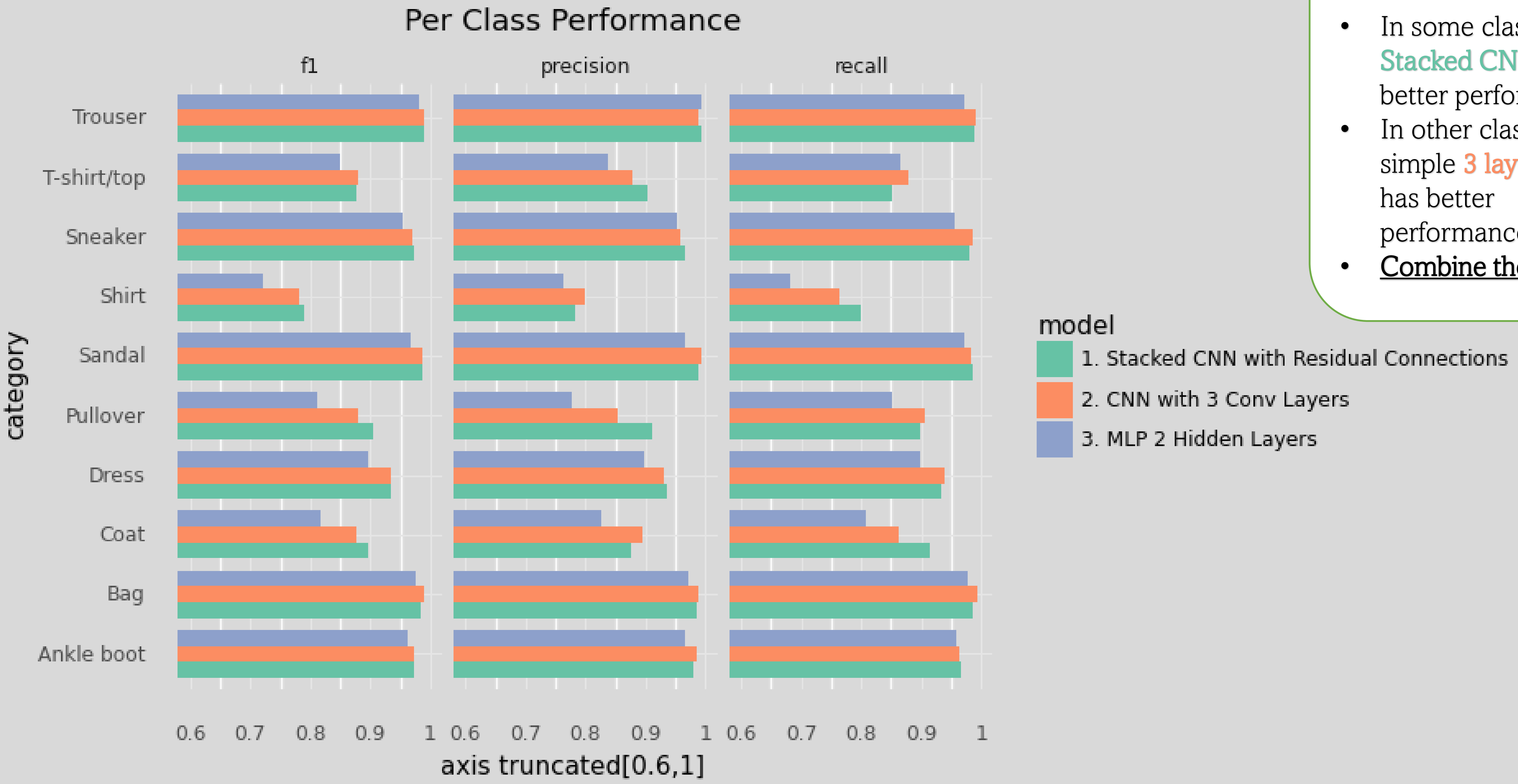
set	accuracy	precision	recall	f1
train	0.972604	0.972719	0.972604	0.972609
val	0.935167	0.935272	0.935167	0.935147
test	**0.931000**	0.931140	0.931000	**0.930924**

Tuned Models



CNNs overperformed the MLP. The Stacked CNN has achieved the best accuracy on the test set

Error Analysis



Looking at the f1 scores:

- In some classes the Stacked CNN has better performance
- In other classes the simple 3 layer CNN has better performance
- <u>Combine them</u>

CNN Ensemble

Argmax (Softmax ($\text{Probabilities}_{\text{CNN1}}$ + $\text{Probabilities}_{\text{CNN2}}$))

Where:

- $\text{Probabilities}_{\text{CNN1}}$ is the SoftMax output of the tuned CNN
- $\text{Probabilities}_{\text{CNN2}}$ is the SoftMax output of the Stacked CNN with Residual connections

The ensemble outperforms the separate performance of its components

model	score
3 Conv Layers	0.9268
3 stacked(x2) Conv layers + Residual Connections	0.9310
Ensemble	**0.9344**