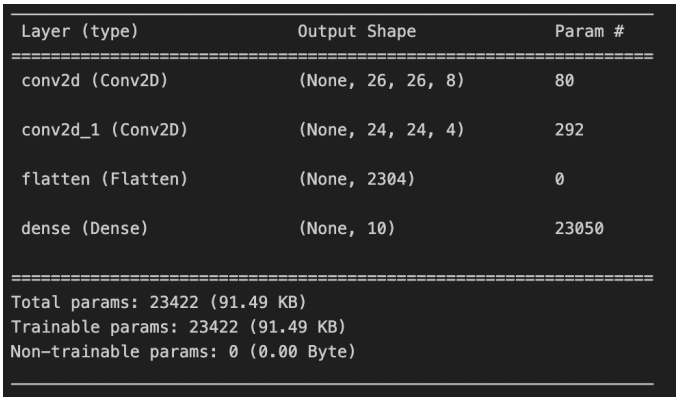
# DL HW2

人工智慧碩二 311505011 黃丰楷

# 1. Using Convolutional Neural Network for Image Recognition

## 1-1 Baseline and Experiments

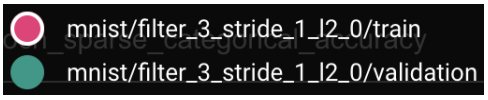
- Model Structure



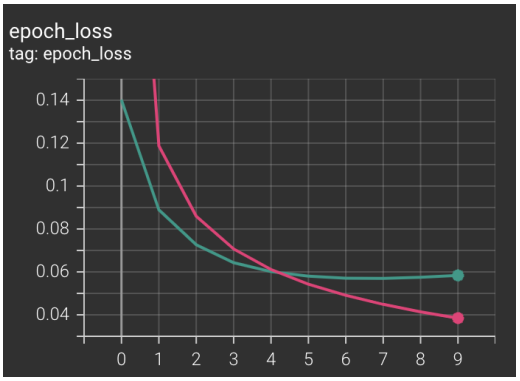
| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 26, 26, 8) | 80 |
| conv2d_1 (Conv2D) | (None, 24, 24, 4) | 292 |
| flatten (Flatten) | (None, 2304) | 0 |
| dense (Dense) | (None, 10) | 23050 |

Total params: 23422 (91.49 KB)
Trainable params: 23422 (91.49 KB)
Non-trainable params: 0 (0.00 Byte)

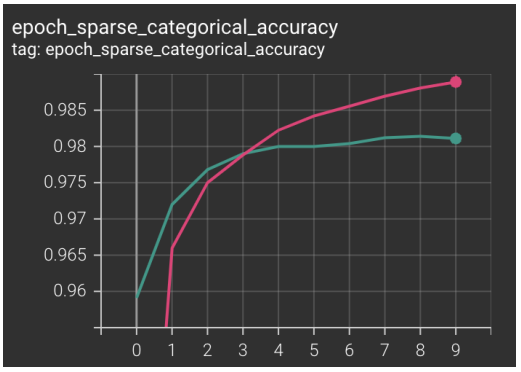
- Implement details
  - Epochs: 10
  - Learning rate: 1e-3
  - Batch size: 128
  - Optimizer: Adam
- Experiments — Baseline
  - Setting
    - Filter size: 3
    - Stride size: 1
    - L2 regularization: 0
  - Legend




- Learning curves

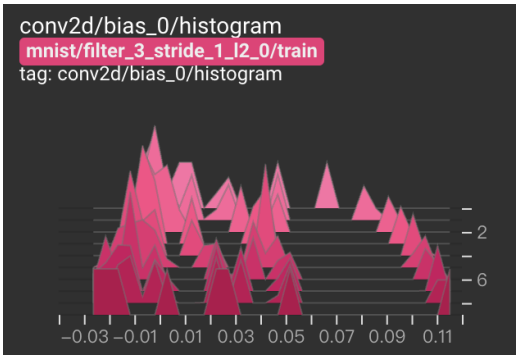



(a) loss curves

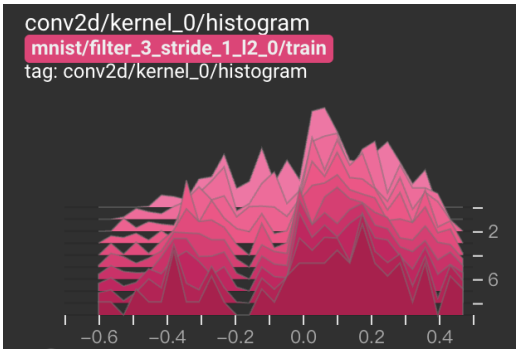



(b) Accuracy curves

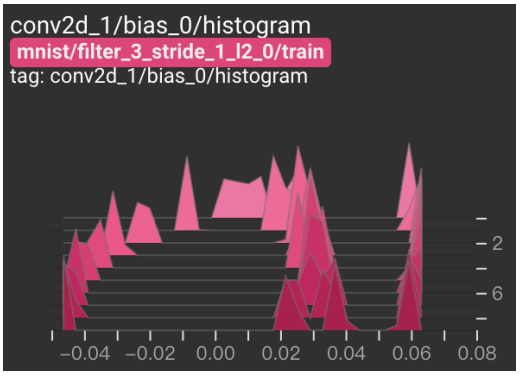
- Histograms



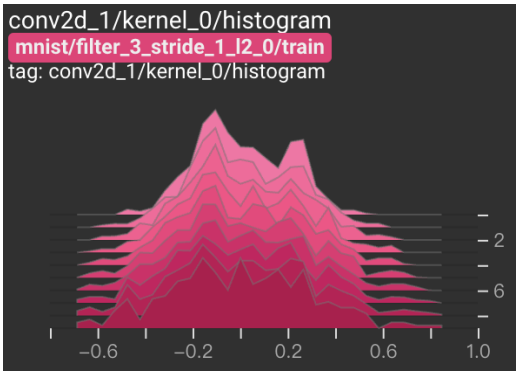

(a) conv2d — bias



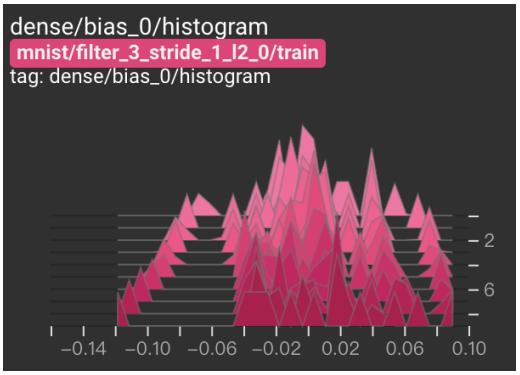

(b) con2d — kernel



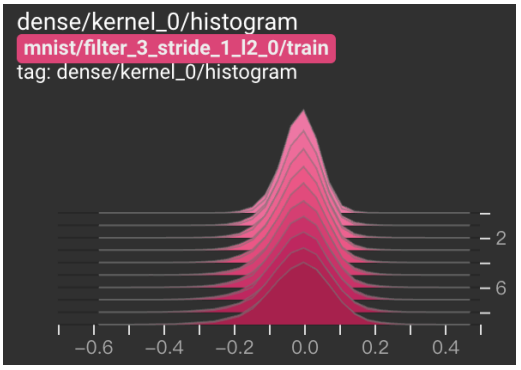

(c) conv2d_1 — bias




(d) conv2d_1 — kernel

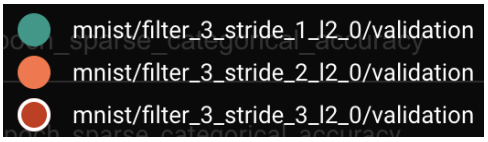



(e) dense — bias

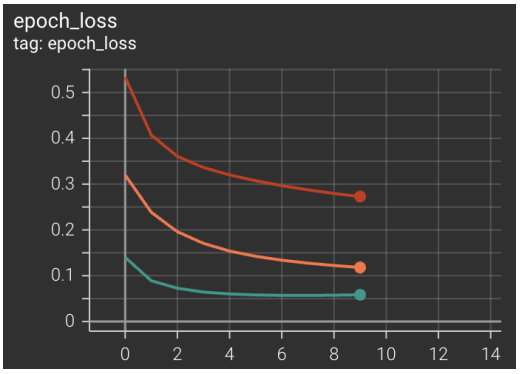



(f) dense — kernel

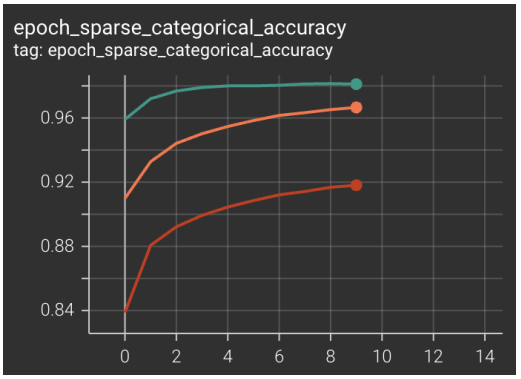
- Experiments — Different stride size
  - Setting
    - Filter size: 3
    - L2 regularization: 0
  - Legend




  - Learning curves

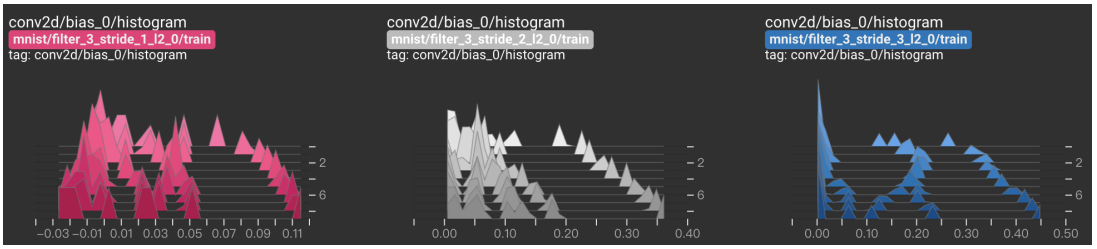



(a) Loss curves

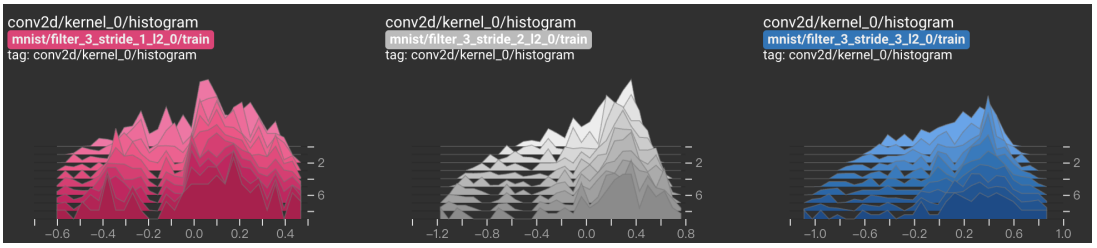



(b) Accuracy curves

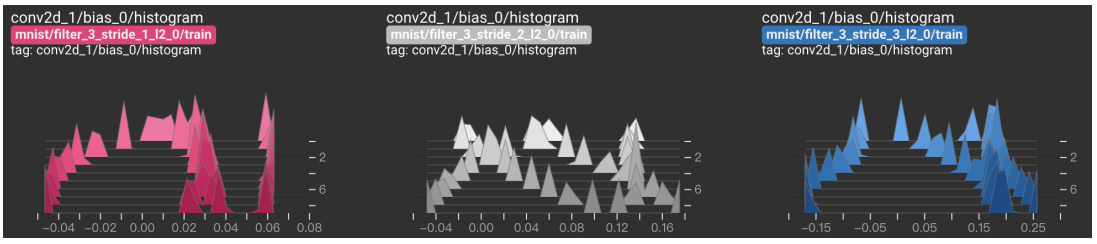
  - Histogram



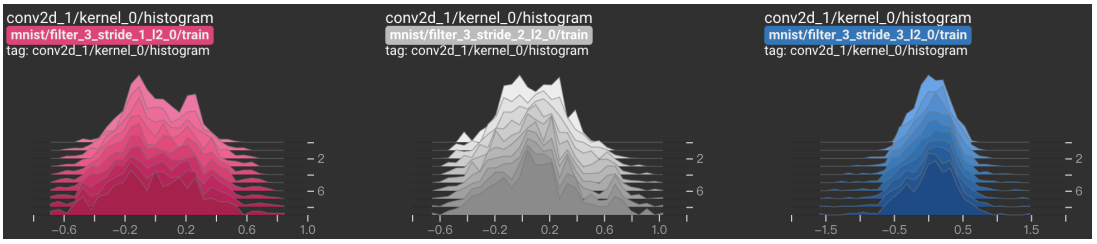

(a) conv2d — bias



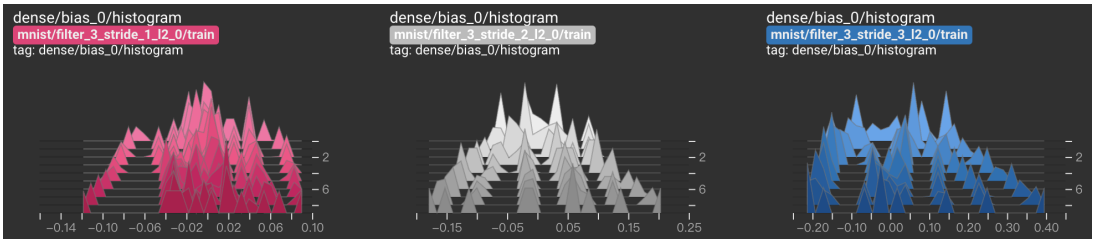

(b) con2d — kernel



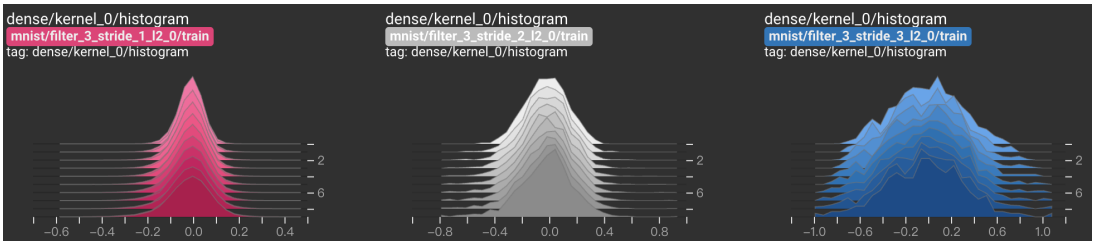

(c) conv2d_1 —bias




(d) conv2d_1 —kernel

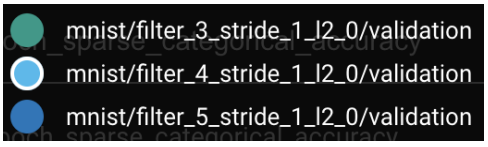



(e) dense —bias

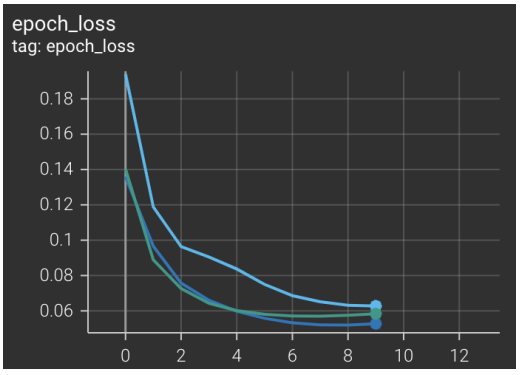



(f) dense —kernel

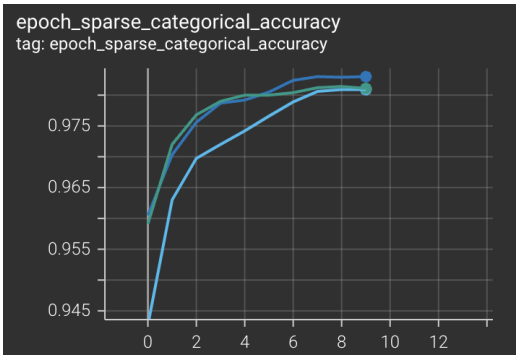
- Experiments — Different filter size
  - Setting
    - Stride size: 1
    - L2 regularization: 0
  - Legend




  - Learning curves

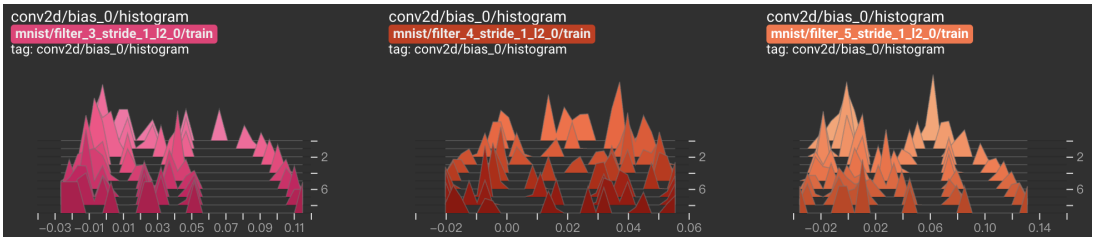



(a) Loss curves

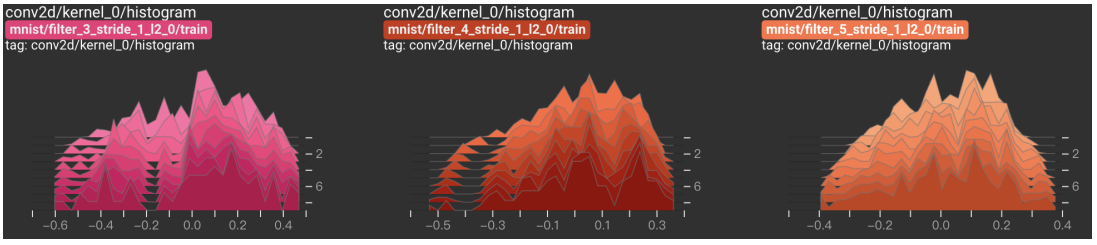



(b) Accuracy curves

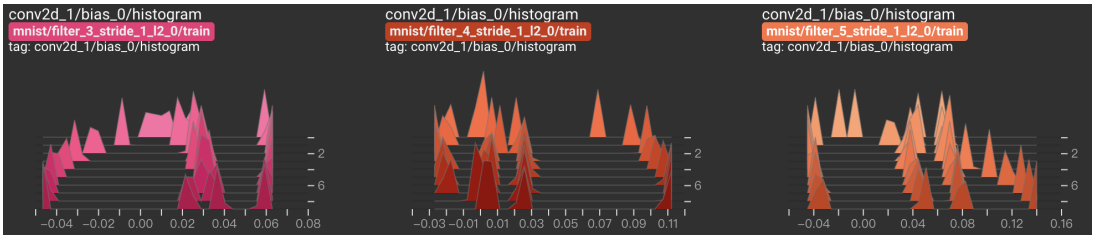
  - Histogram



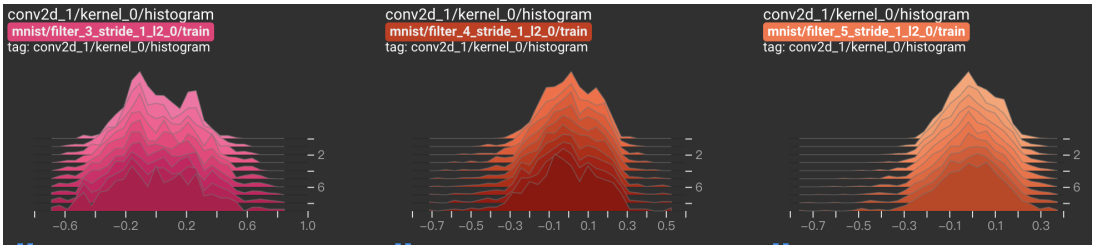

(a) conv2d — bias



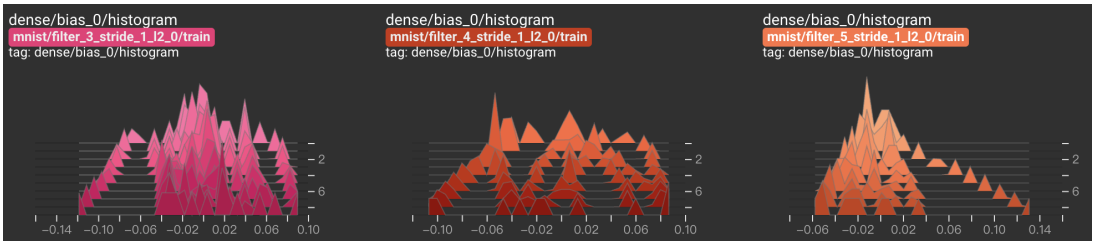

(b) con2d — kernel



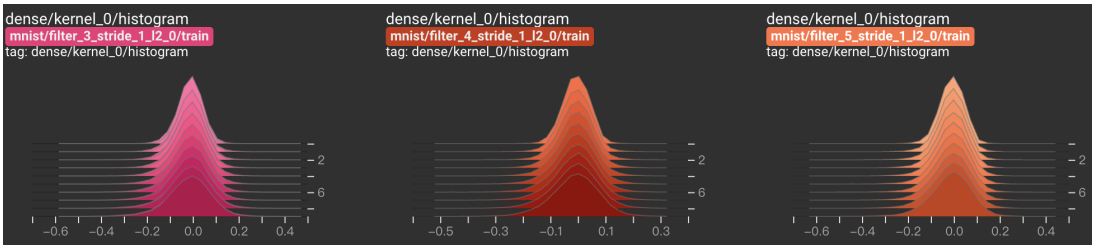

(c) conv2d_1 — bias




(d) conv2d_1 — kernel




(e) dense — bias




(f) dense — kernel

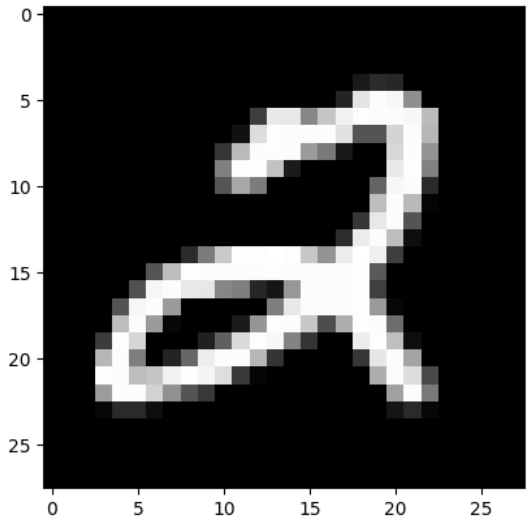
- Experiment — Conclusion

  As we can see in the following table, the bigger the filter size is, the higher the validation accuracy becomes. Since the respective filed increase when the kernel of CNN is larger, the model can see bigger area. However, the bigger the stride size is, the lower the validation accuracy becomes. Because when the stride size increase, the model may loss some information during convolution operation.

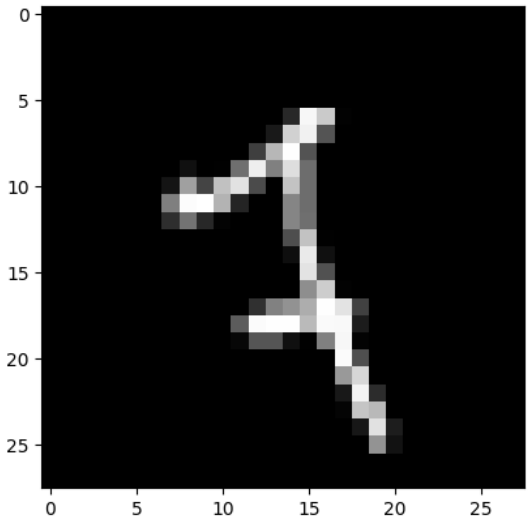
| Filter size | Stride size | Val accuracy |
|---|---|---|
| 3 | 1 | 0.981 |
| 3 | 2 | 0.967 |
| 3 | 3 | 0.918 |
| 4 | 1 | 0.981 |
| 5 | 1 | 0.983 |

# 1-2 Classified vs. Miss-classified

- Figures



(a) Classified (prediction: 2, label: 2)



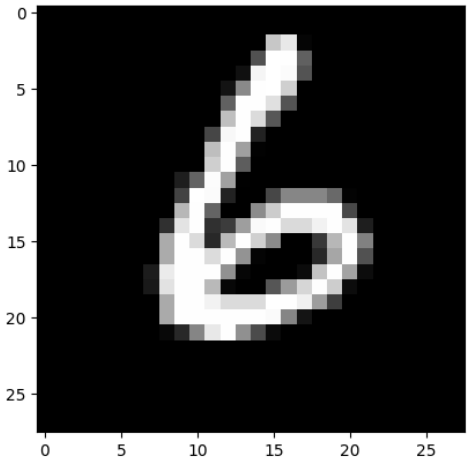
(b) Miss-classified (prediction: 2, label: 7)

- Discussion

  In these two cases, the predictions are both 2, but the label are 2 and 7, respectively. In figure (b), the pattern is a little bit weird unlike the standard 7, which contains a horizontal line in the middle of 7. Therefore, it is a hard case for model to learn because it is similar to 2.

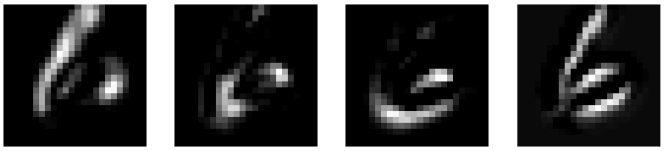
## 1-3 Feature maps

- Original image




- Feature maps in conv2d

- Feature maps in conv2d_1

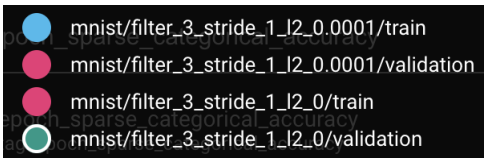


- Discussion

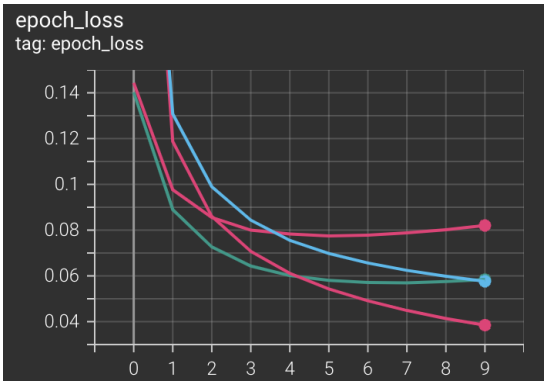
First, the channel size of feature maps in deeper layer is designed smaller. Second, deeper layer can extract high-level feature maps.

## 1-4 L2 Regularization

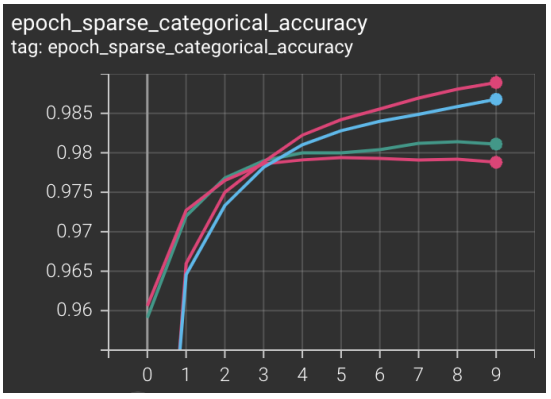
- Setting
  - Stride size: 3
  - Filter size: 1
- Legend




- Learning curves

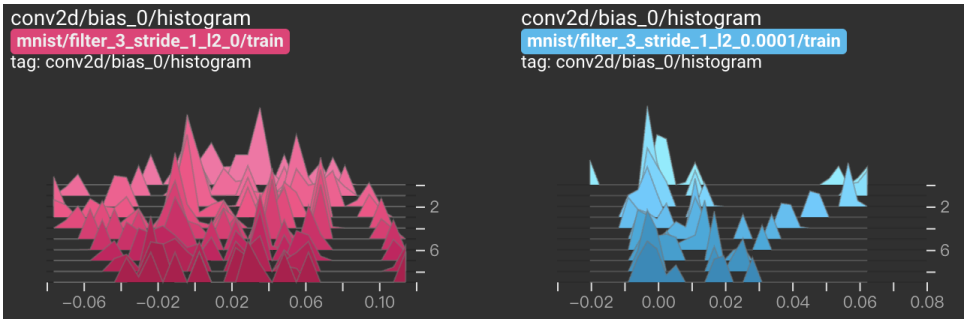



(a) Loss curves

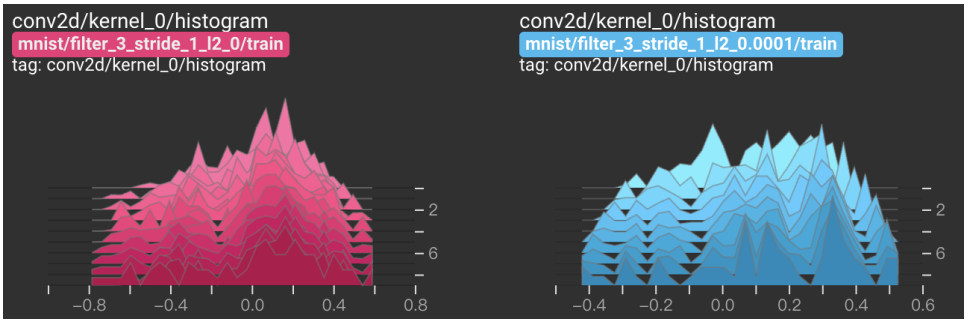



(b) Accuracy curves

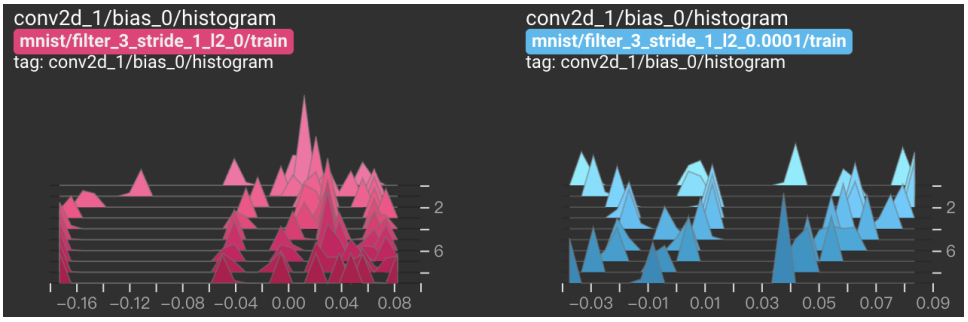
- Histogram



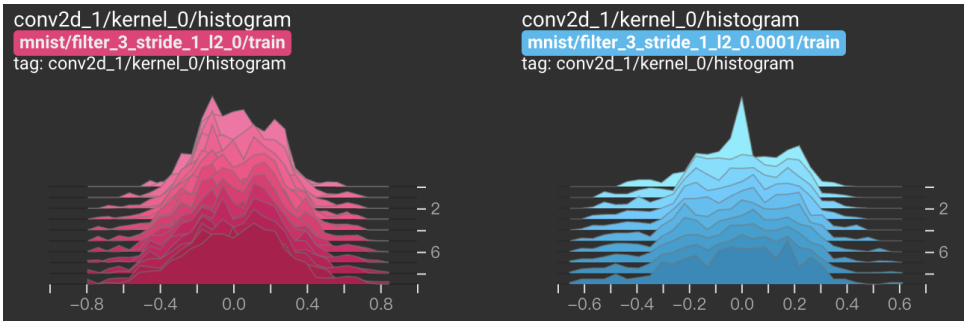

(a) conv2d — bias



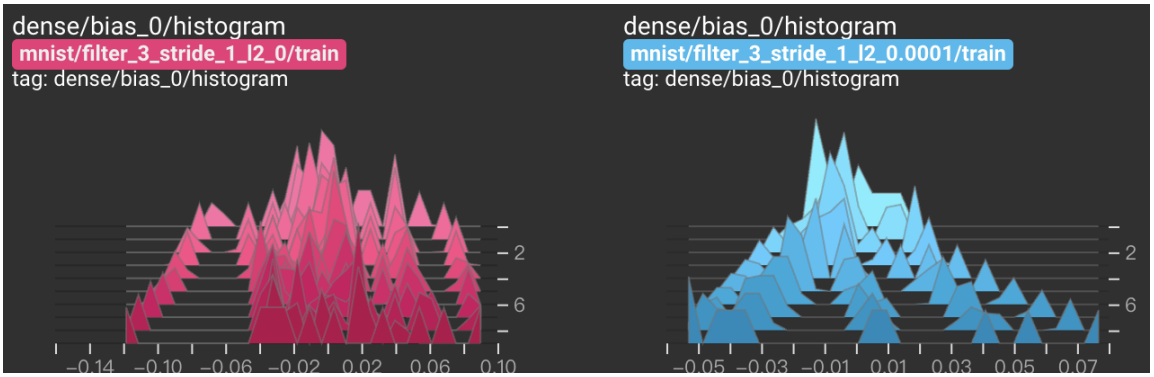

(b) con2d — kernel



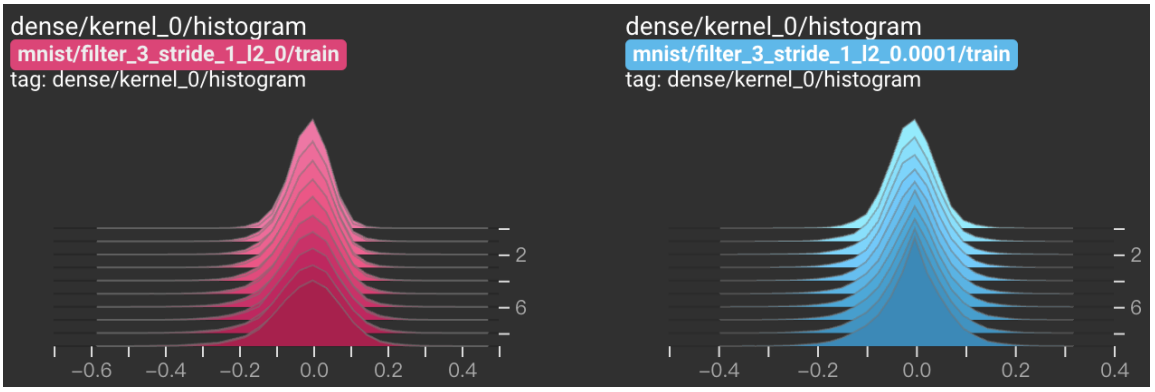

(c) conv2d_1 — bias




(d) conv2d_1 — kernel




(e) dense — bias




(f) dense — kernel

- Discussion

  As we can see in the following table, after adding l2 regularization, the performance drops a little bit. Furthermore, in the histograms, the absolute value of  parameter distributions of the one with l2 regularization are smaller because the regularization constraint. The performance does not improve after adding regularization because there is no overfitting phenomenon in this case.

| Train/Val | L2 Regularization | Val accuracy |
|---|---|---|
| Train | No | 0.989 |
| Val | No | 0.981 |
| Train | Yes | 0.987 |
| Val | Yes | 0.979 |

# 2. Preprocessing Before Using Convolutional Neural Network for Image Recognition

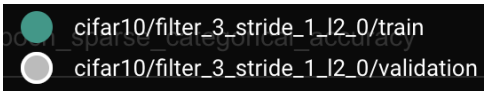
## 2-1 Baseline and Experiments

- Model Structure

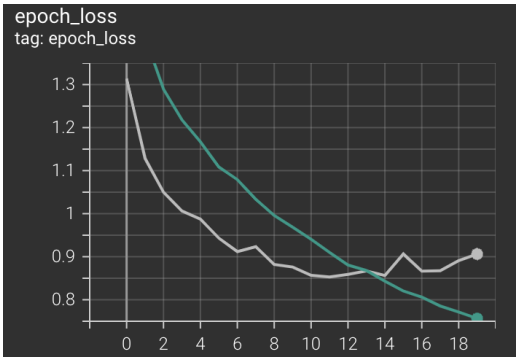
| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 32, 32, 64) | 1792 |
| max_pooling2d (MaxPooling2D) | (None, 16, 16, 64) | 0 |
| conv2d_1 (Conv2D) | (None, 16, 16, 128) | 73856 |
| max_pooling2d_1 (MaxPooling2D) | (None, 8, 8, 128) | 0 |
| flatten (Flatten) | (None, 8192) | 0 |
| dense (Dense) | (None, 64) | 524352 |
| dropout (Dropout) | (None, 64) | 0 |
| dense_1 (Dense) | (None, 10) | 650 |

Total params: 600650 (2.29 MB)
Trainable params: 600650 (2.29 MB)
Non-trainable params: 0 (0.00 Byte)

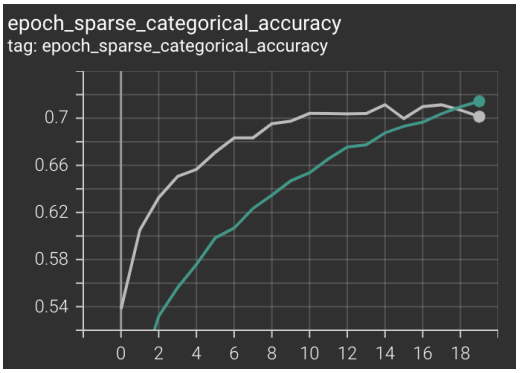
- Implement details
  - Epochs: 20
  - Learning rate: 1e-3
  - Batch size: 64
  - Optimizer: Adam
- Experiments — Baseline
  - Setting
    - Filter size: 3
    - Stride size: 1
    - L2 regularization: 0
  - Legend




  - Learning curves

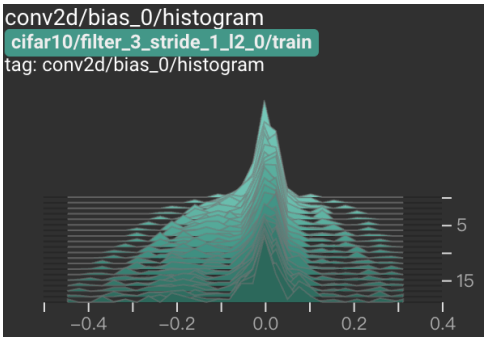



(a) loss curves

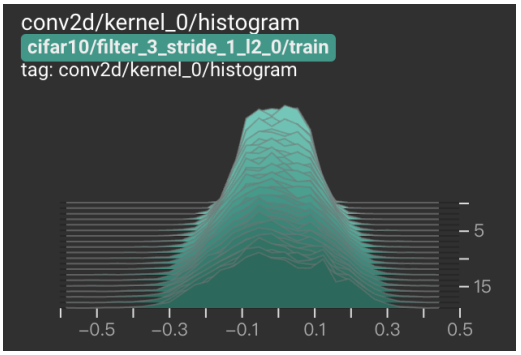



(b) Accuracy curves

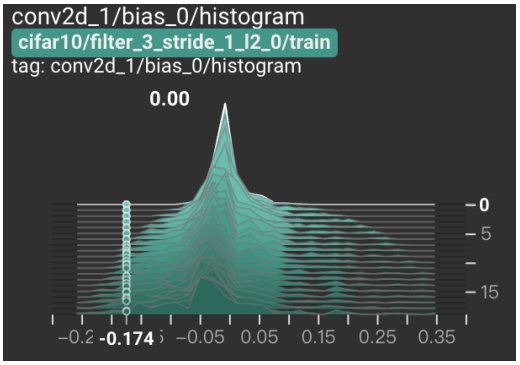
- Histograms



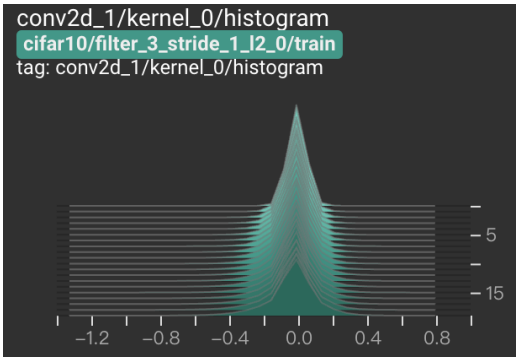

(a) conv2d — bias



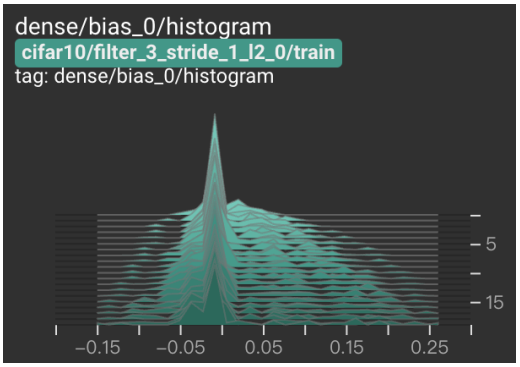

(b) con2d — kernel



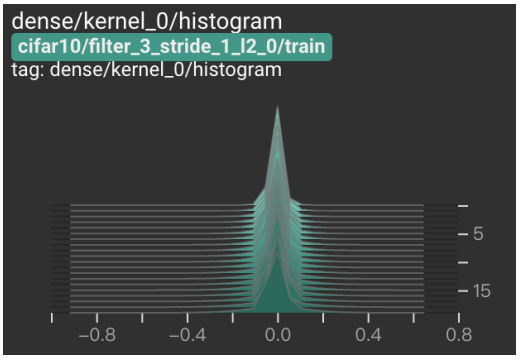

(c) conv2d_1 — bias



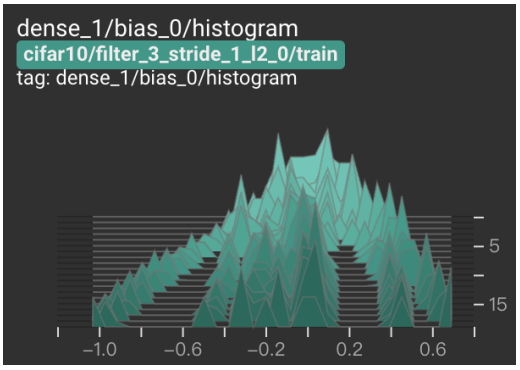

(d) conv2d_1 — kernel



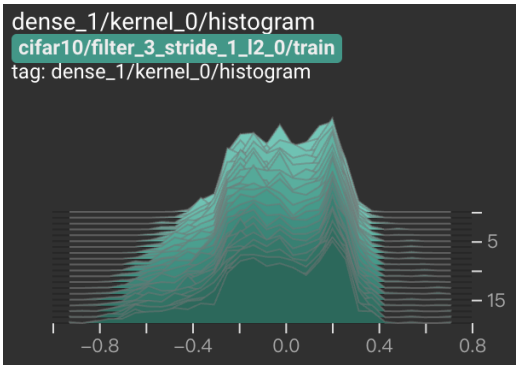

(e) dense — bias




(f) desne — kernel

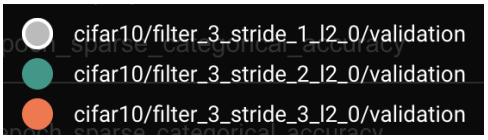



(g) dense_1 — bias

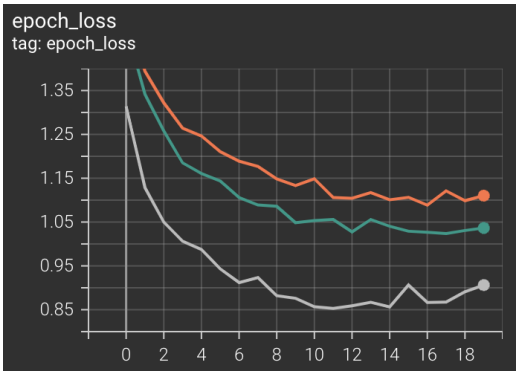



(h) desne_1 — kernel

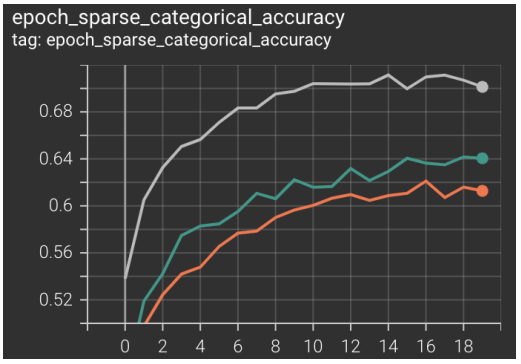
- Experiments — Different stride size
  - Setting
    - Filter size: 3
    - L2 regularization: 0
  - Legend




  - Learning curves

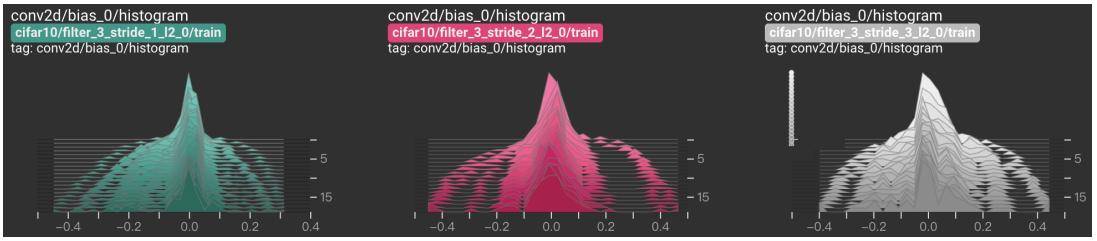



(a) Loss curves

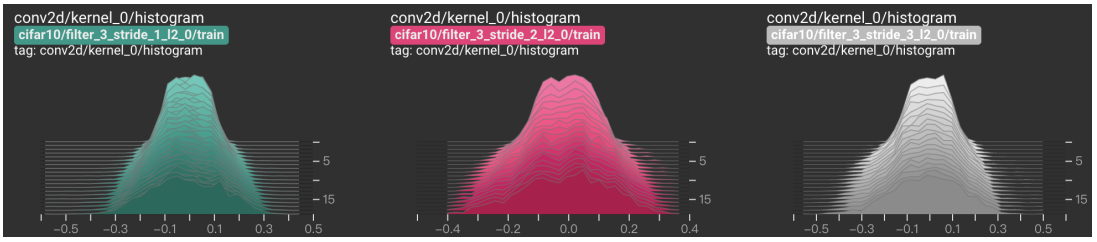



(b) Accuracy curves

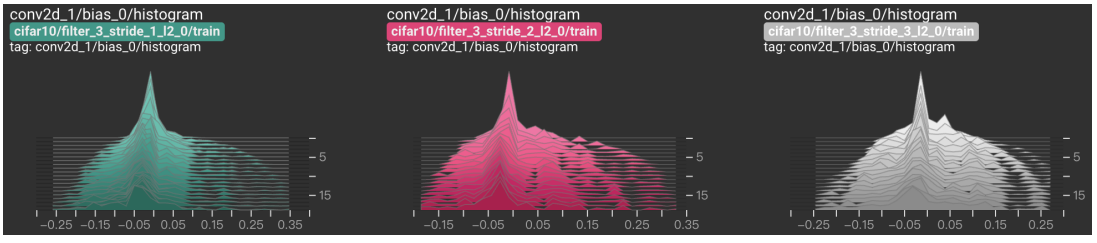
- Histogram



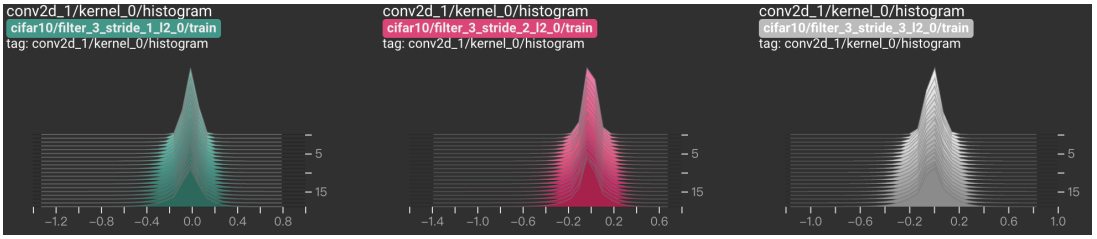

(a) conv2d — bias



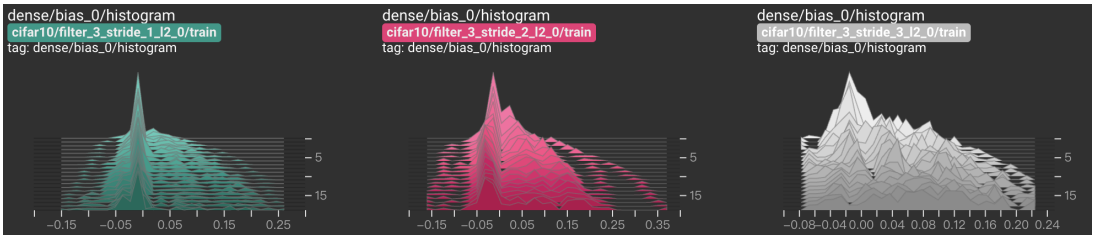

(b) con2d — kernel



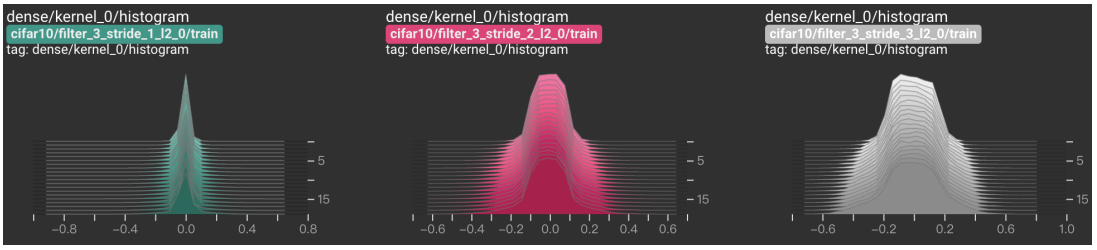

(c) conv2d_1 —bias



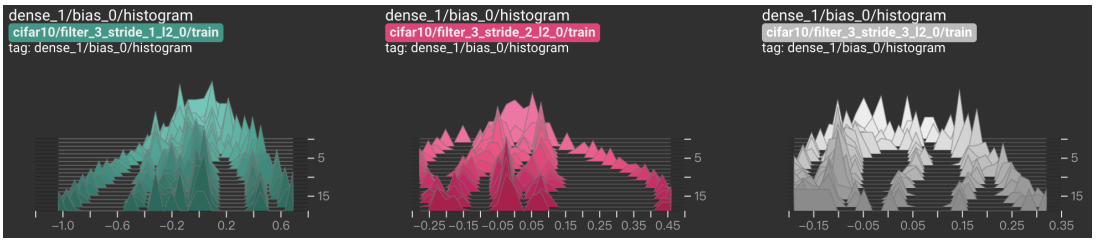

(d) conv2d_1 —kernel



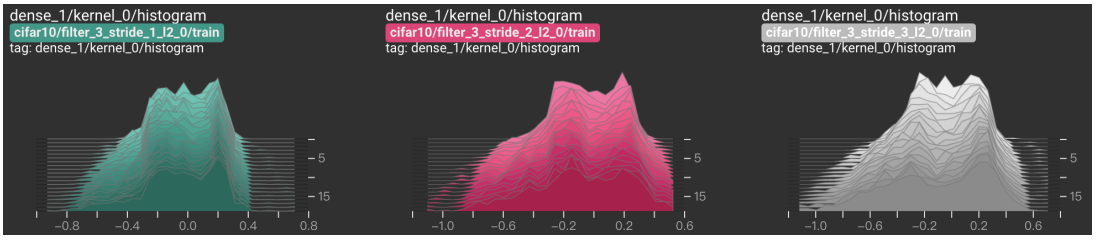

(e) dense —bias




(f) dense —kernel




(g) dense_1 —bias

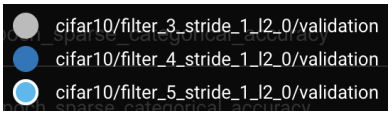



(h) dense_1 —kernel

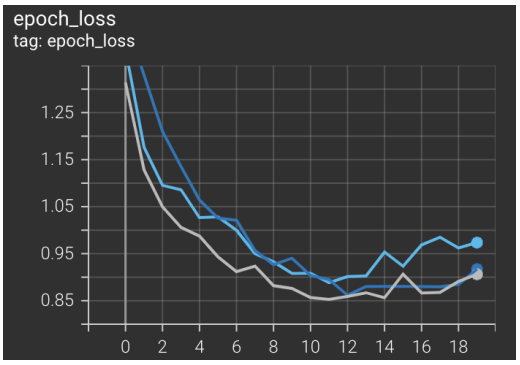
- Experiments — Different filter size
  - Setting

- Stride size: 1
- L2 regularization: 0

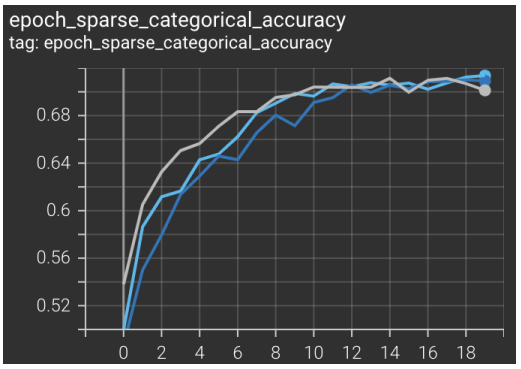
- Legend




- Learning curves

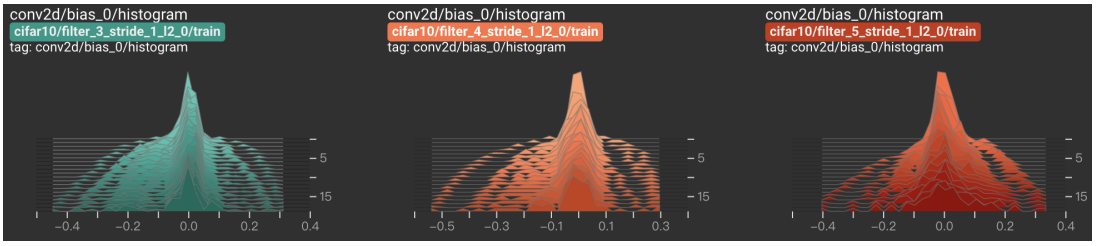



(a) Loss curves

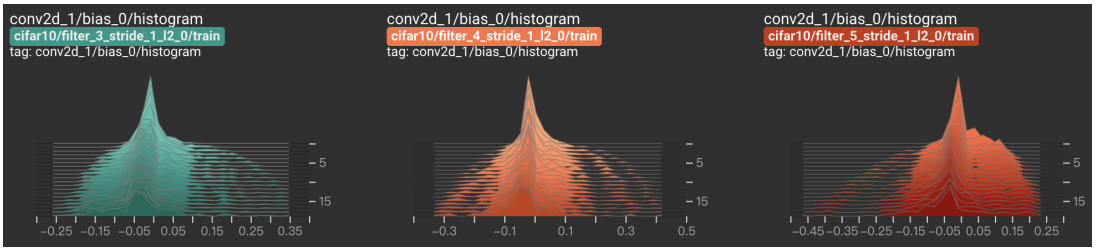



(b) Accuracy curves

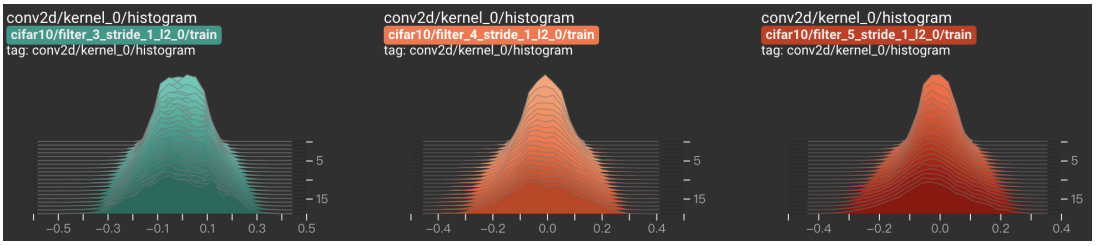
- Histogram



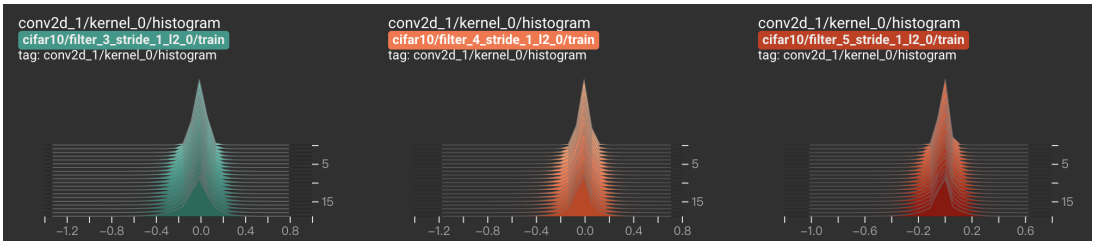

(a) conv2d — bias



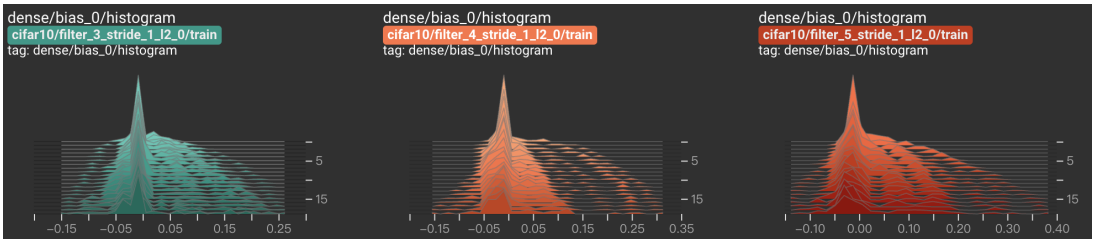

(b) con2d — kernel



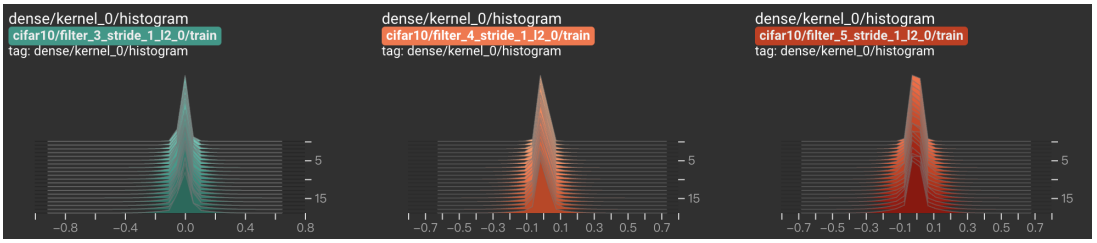

(c) conv2d_1 —bias



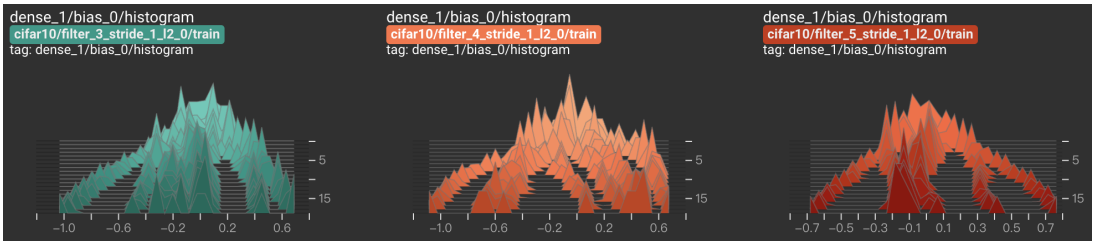

(d) conv2d_1 —kernel



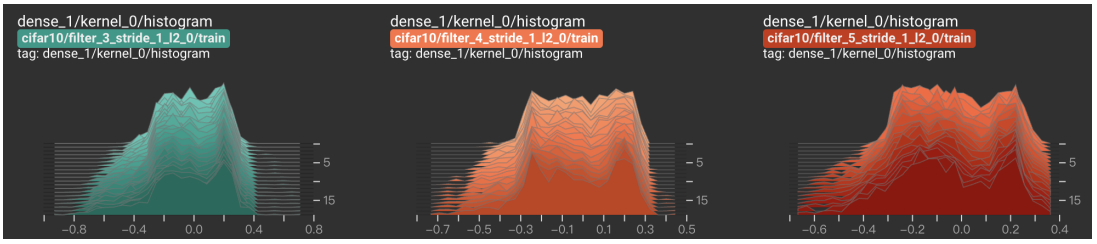

(e) dense —bias




(f) dense —kernel




(g) dense_1 —bias




(h) dense_1 —kernel

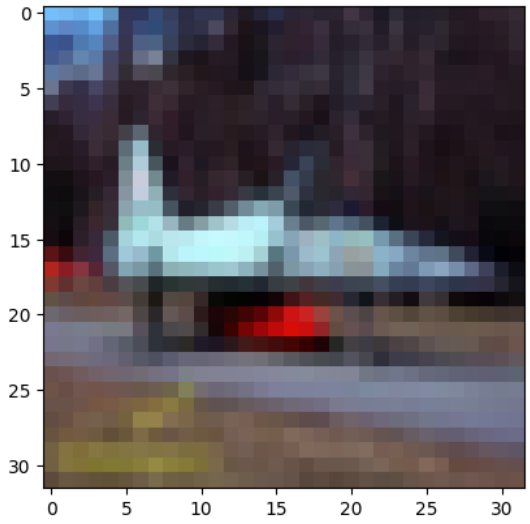
- Experiment — Conclusion

  The result aligns with the mnist dataset. As illustrated in the table below, it's evident that as the filter size increases, so does the validation accuracy. This can be attributed to the fact that with a larger kernel in the CNN, the model can observe a wider area due to the increased receptive field. Conversely, when it comes to the stride size, a larger value leads to a decline in validation accuracy. This is because an increased stride size during convolution may result in the loss of crucial information.

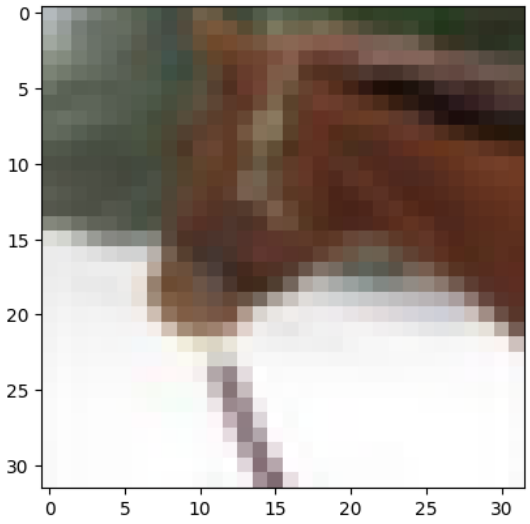
| Filter size | Stride size | Val accuracy |
|---|---|---|
| 3 | 1 | 0.701 |
| 3 | 2 | 0.641 |
| 3 | 3 | 0.613 |
| 4 | 1 | 0.709 |
| 5 | 1 | 0.714 |

## 2-2 Classified vs. Miss-classified

- Figures



(a) Classified (prediction: airplane, label: airplane)



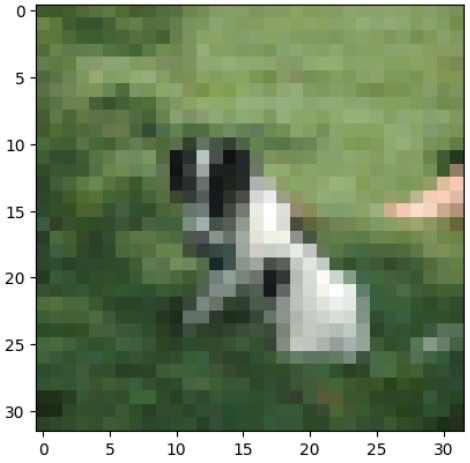
(b) Miss-classified (prediction: dog, label: horse)

- Discussion

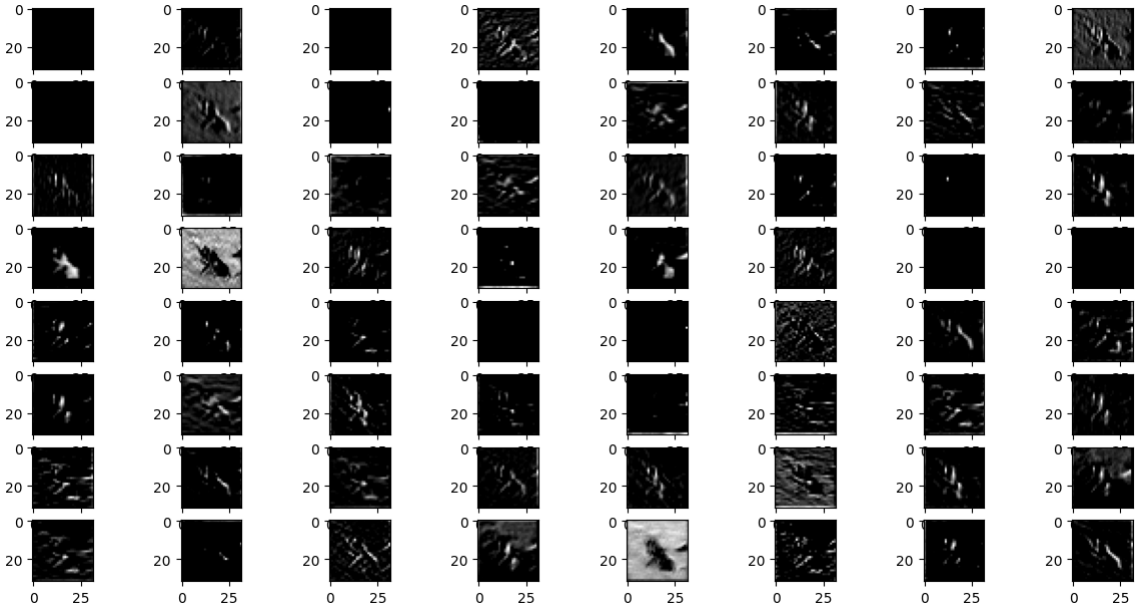
  In figure (b), the horse is classified as dog. The reason of mis-classifying may be that the image only contains the head of the horse, which losses the information of its body. Further more, the color of the horse are very similar to a dog, so that this horse is easily to be classified as a dog.

## 2-3 Feature maps

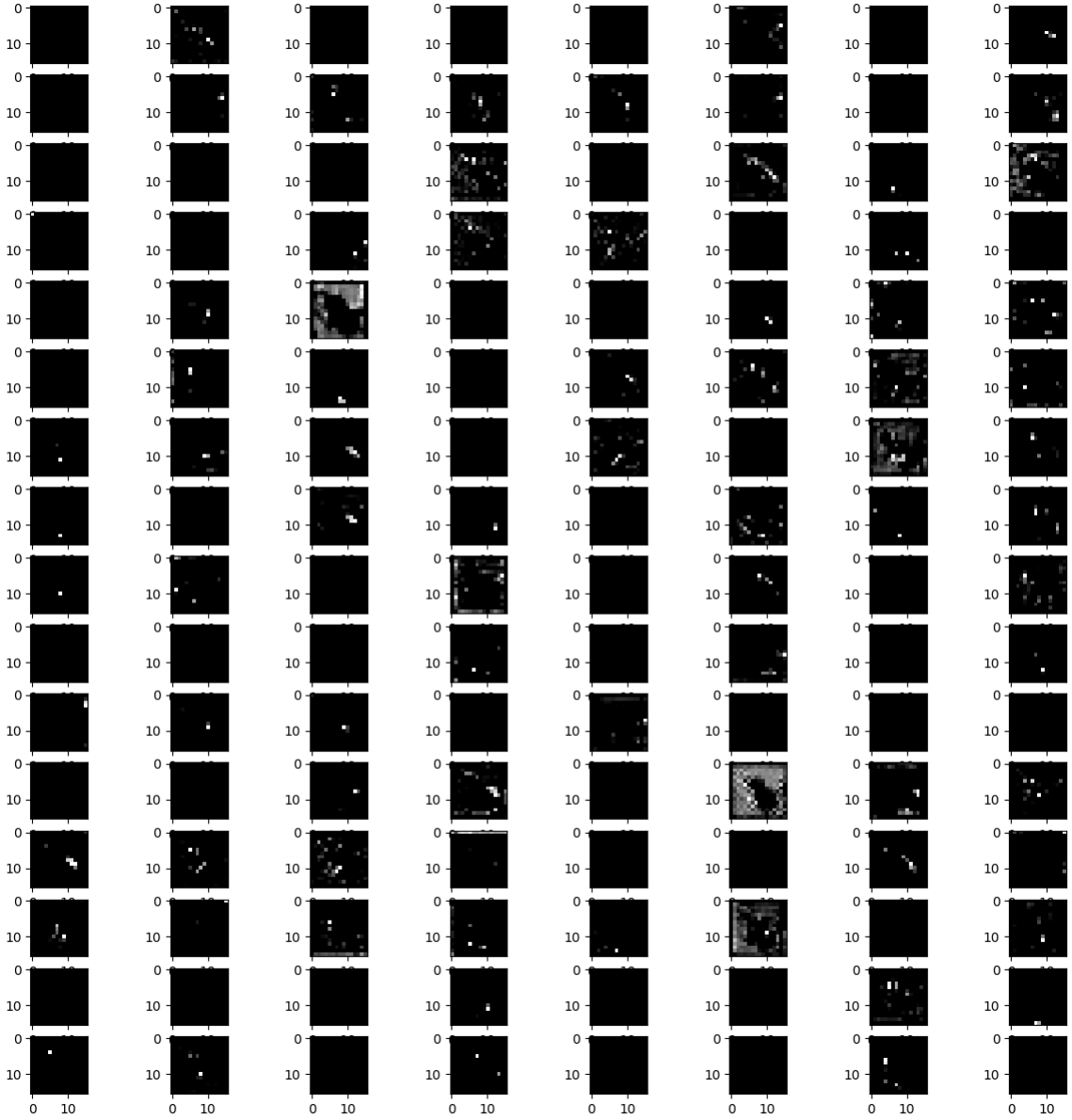
- Original image



- Feature maps in conv2d



- Feature maps in conv2d_1

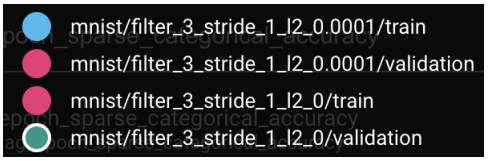


- Discussion

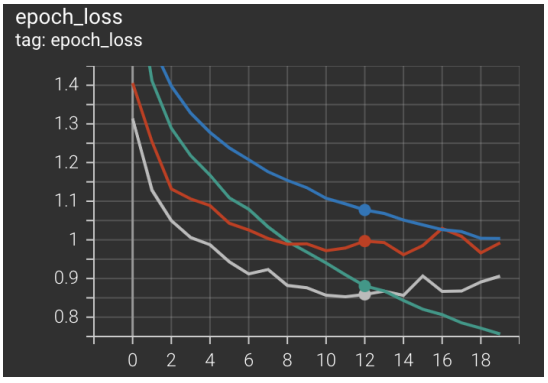
  Firstly, the channel size of feature maps in deeper layers is intentionally made larger. Secondly, deeper layers have the capability to extract high-level feature maps, which possess larger receptive fields.

## 2-4 L2 Regularization

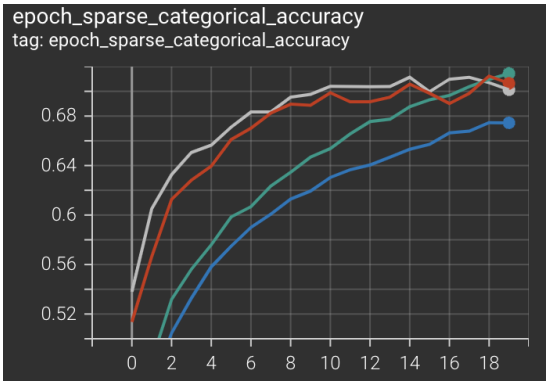
- Setting
  - Stride size: 3
  - Filter size: 1
- Legend




- Learning curves

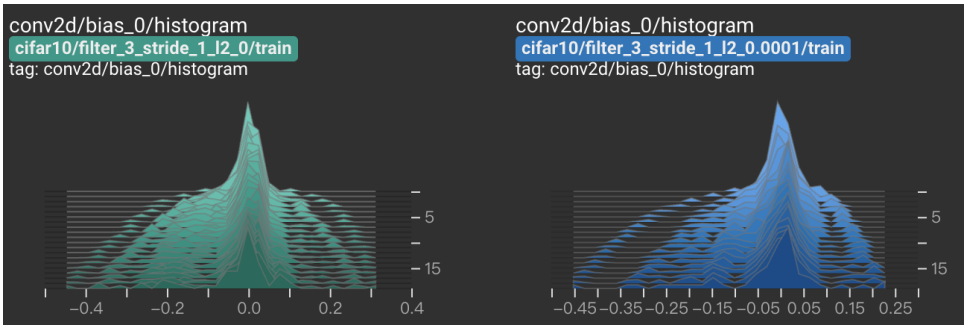



(a) Loss curves

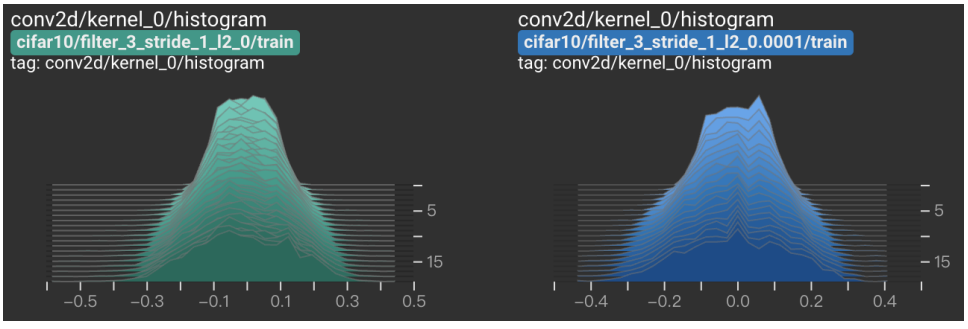



(b) Accuracy curves

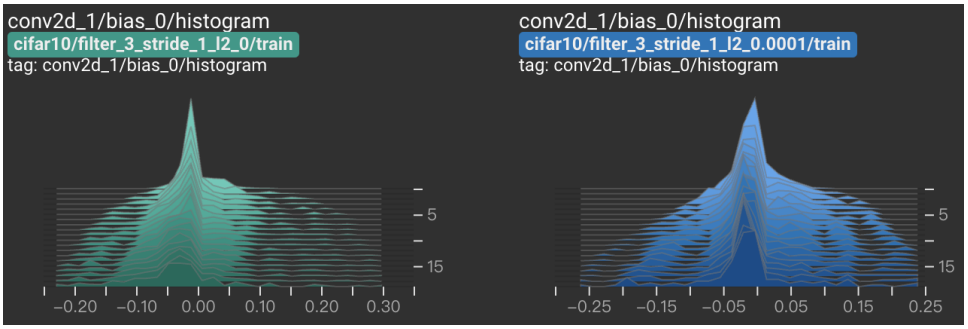
- Histogram



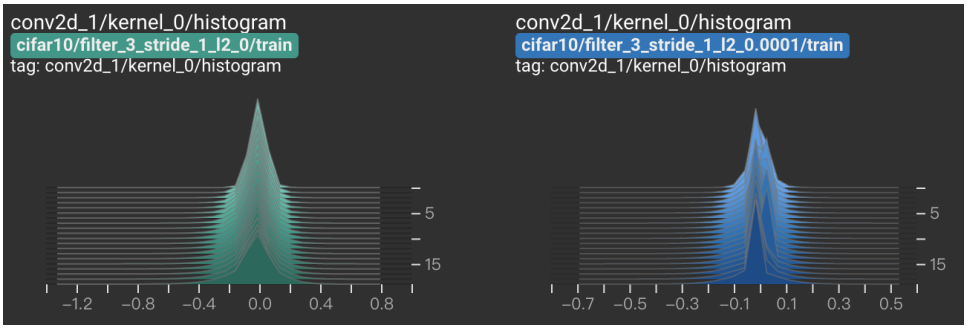

(a) conv2d — bias



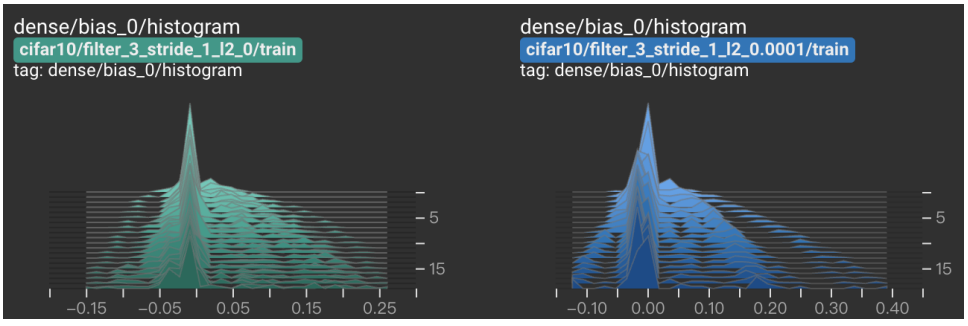

(b) con2d — kernel



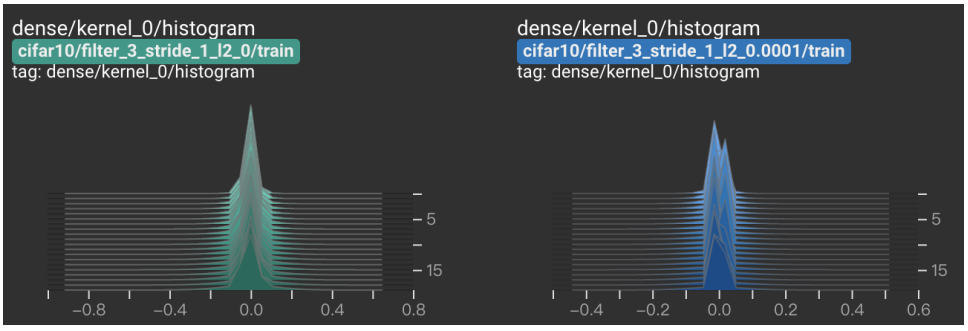

(c) con2d_1 — bias



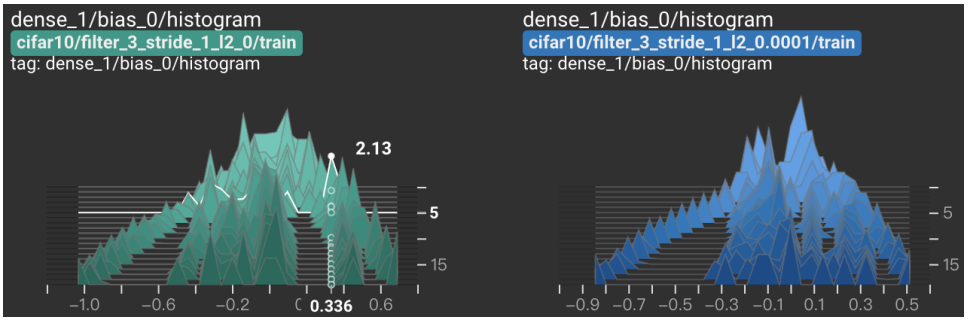

(d) conv2d_1 —kernel



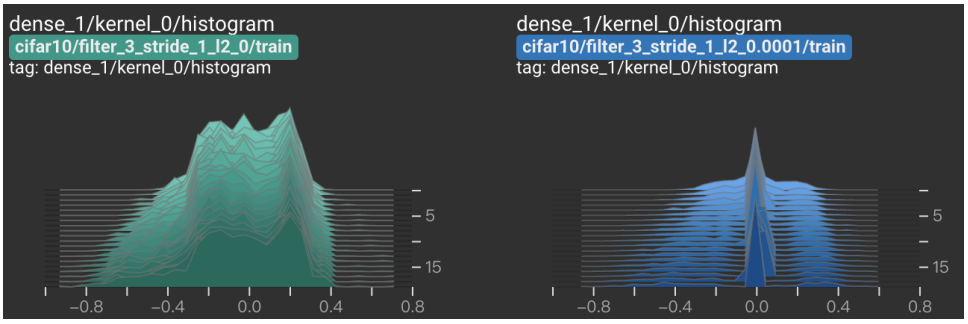

(e) dense —bias




(f) dense —kernel




(g) dense_1 —bias




(h) dense_1 —kernel

- Discussion

  As depicted in the table below, the introduction of l2 regularization leads to a slight decrease in training accuracy. Conversely, the validation accuracy demonstrates an improvement with the incorporation of l2 regularization. Examining the parameter distributions in the histograms, it is evident that the absolute values are smaller in the case with l2 regularization, indicating the effect of the regularization constraint. The enhancement in validation performance suggests a mitigation of potential overfitting. In summary, the inclusion of l2 regularization results in a decrease in training accuracy due to the imposed constraint, while simultaneously enhancing the model's generalizability, ultimately leading to improved validation accuracy.

| Train/Val | L2 Regularization | Val accuracy |
|---|---|---|
| Train | No | 0.714 |
| Val | No | 0.701 |
| Train | Yes | 0.674 |
| Val | Yes | 0.706 |

# 2-5 Preprocessing

- Operations

    - Splitting training/validation set
    - Shuffle: adding randomness to training
    - Normalization: normalize the input images to [0,1]
    - Batchisation: batch the training/validation images.
- Code

```
train_data = train_data.shuffle(train_num)
train_data = train_data.map(map_func=parse_fn, num_parallel_calls=AUTOTUNE)
train_data = train_data.batch(BATCH_SIZE)
train_data = train_data.cache()
train_data = train_data.prefetch(tf.data.AUTOTUNE)


test_data = test_data.map(map_func=parse_fn, num_parallel_calls=AUTOTUNE)
test_data = test_data.batch(BATCH_SIZE)
test_data = test_data.cache()
test_data = test_data.prefetch(tf.data.AUTOTUNE)
```