2021.02.04

Collision Avoidance (DNN)

DGIST

Jaewoo Park

Collision Avoidance

Simplify the dimension

- **Input features**: 32×32**×3** (3072) → 32×32**×1** (1024)

Change the DNN model

- Variable hidden layers
- Batch normalization
- Dropout (same value for every hidden layers)

Change the DNN model

```
INPUT_SIZE = IMAGE_HEIGHT * IMAGE_WIDTH * IMAGE_CHANNEL

class DNN(nn.Module):

    __slots__ = "__model"

    def __init__(self, input_dim=INPUT_SIZE, output_dim=2, hidden_dims=(128, 64, 32), do_batch_normal=True, dropout=0):

        super(DNN, self).__init__()

        dims_list = (input_dim, *hidden_dims)
        model_components = []

        # hidden layers
        for i in range(1, len(dims_list)):
            current_input_dim = dims_list[i-1]
            current_output_dim = dims_list[i]
            model_components.append(nn.Linear(current_input_dim, current_output_dim))

            if do_batch_normal == True:
                model_components.append(nn.BatchNorm1d(current_output_dim))

            model_components.append(nn.ReLU())

            if dropout > 0:
                model_components.append(nn.Dropout(dropout))

        # output layer
        output_layer = nn.Linear(dims_list[-1], output_dim)
        model_components.append(output_layer)
        model_components.append(nn.Softmax(dim=1))

        # make DNN model
        self.__model = nn.Sequential(*model_components)

    def forward(self, x):
        return self.__model(x)
```

Collision Avoidance

Train Result

- **Subject**: Effect of the *dropout*
- **Hidden Layers**: 1280, 720, 160
- **Hyper parameters**
  Epochs: 30, Learning rate: 0.001, Momentum: 0.9, L2 constant: 1e-5
- **Normalization**: Batch normalization only

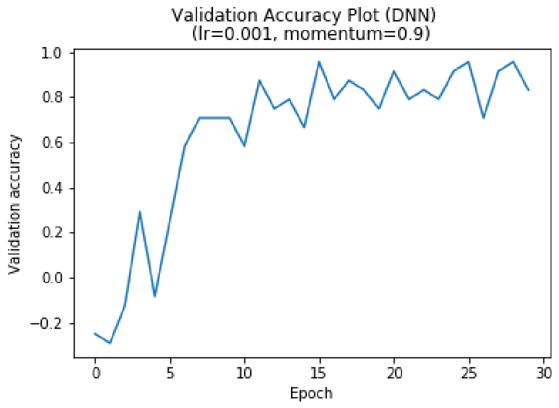
Figure	Dropout	Validation Accuracy	Test Accuracy*
(a)	0.0	0.958	0.875 (21/24)
(b)	0.2	0.750	0.875 (21/24)
(c)	0.4	0.542	0.792 (19/24)

*(Correct test data/Total test data)

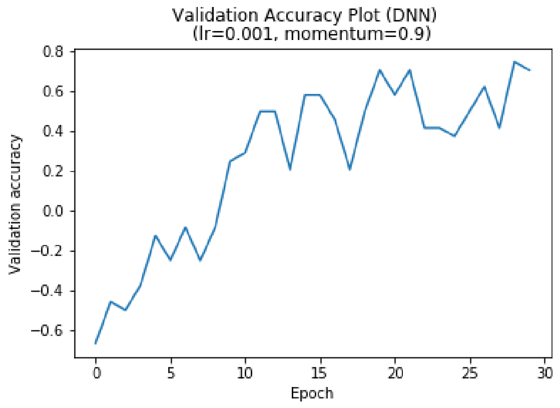
Collision Avoidance

Train Result

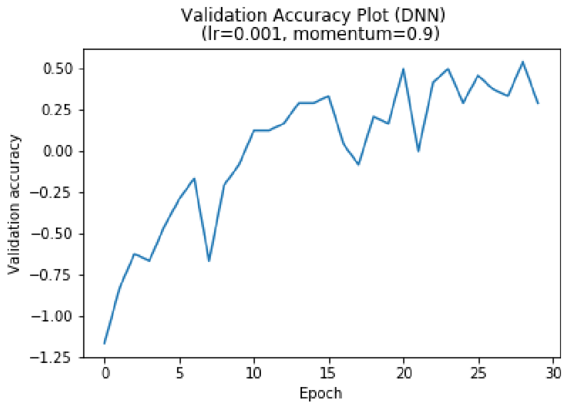
- **Subject**: Effect of the *dropout*



(a)



(b)



(c)

Figure	Dropout	Validation Accuracy	Test Accuracy*
(a)	0.0	0.958	0.875 (21/24)
(b)	0.2	0.750	0.875 (21/24)
(c)	0.4	0.542	0.792 (19/24)

*(Correct test data/Total test data)

Collision Avoidance

Train Result

- **Subject**: Effect of the *dropout*

Dropout is not useful
(In this situation...)

Collision Avoidance

Train Result

- **Subject**: Effect of the hidden layers
- **Hyper parameters**
  Epochs: 30, Learning rate: 0.001, Momentum: 0.9, L2 constant: 1e-5
- **Normalization**: Batch normalization only
- **Dropout**: None (0)

Figure	Hidden layers	Validation Accuracy	Test Accuracy*
(a)	1280, 720, 160	0.958	0.875 (21/24)
(d)	128, 72, 16	0.833	0.917 (22/24)
(e)	128, 64, 16	0.875	0.875 (22/24)
(f)	128, 64, 16	0.917	0.958 (23/24)

Epochs: 60

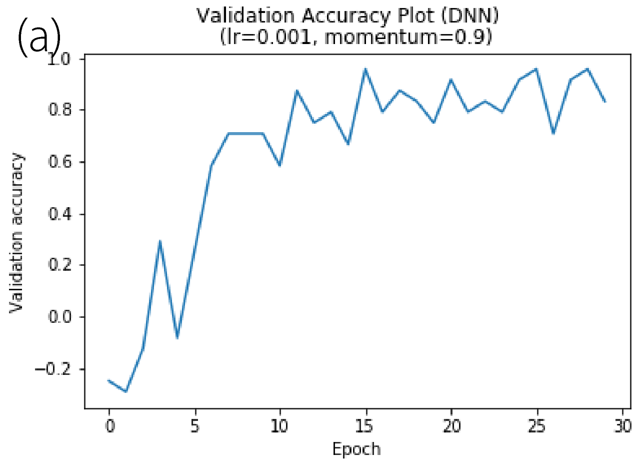
*(Correct test data/Total test data)

Collision Avoidance

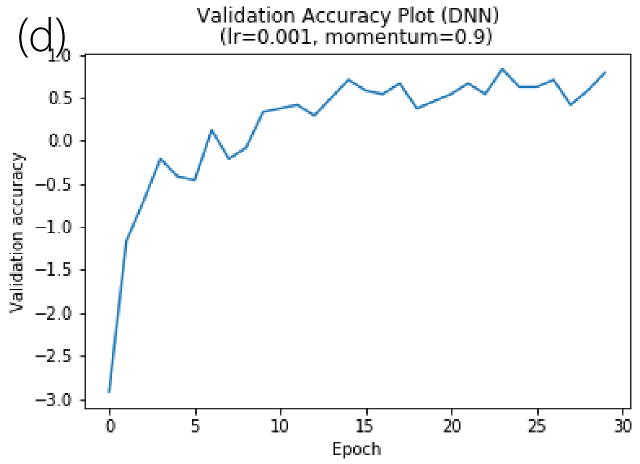
Train Result

- **Subject**: Effect of the hidden layers

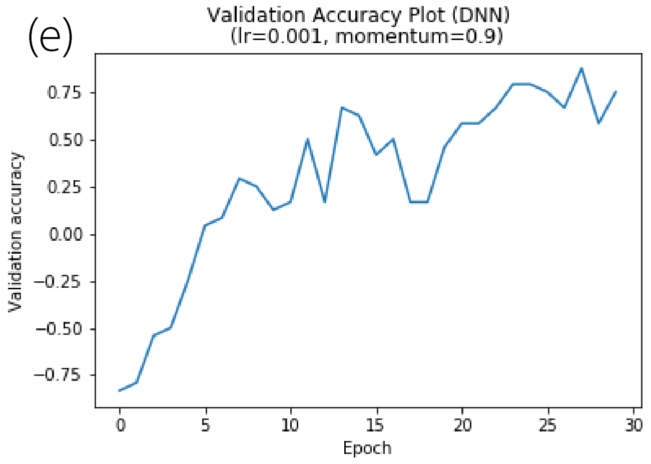
(a)



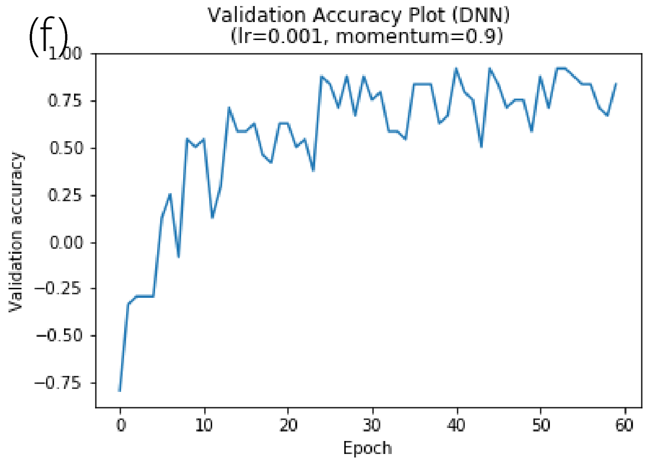
(d)



(e)



(f)



Collision Avoidance

Train Result

- **Subject**: Effect of the hidden layers

1. **2~3 digits are enough**
2. **We need more than 30 epochs**

Collision Avoidance

Train Result

- **Subject**: Effect of the L2 constant
- **Hidden Layers**: 128, 64, 16
- **Hyper parameters**
  Epochs: 30, Learning rate: 0.001, Momentum: 0.9
- **Normalization**: Batch normalization only
- **Dropout**: None (0)

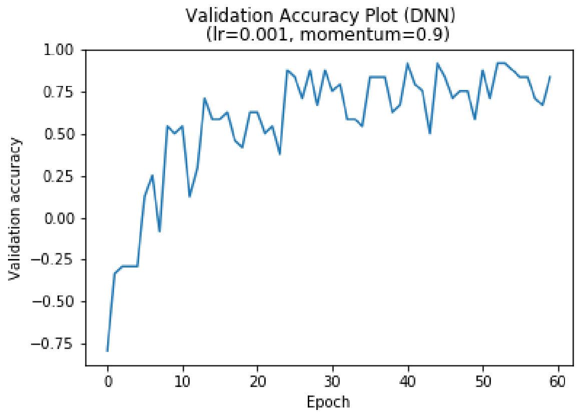
Figure	L2 constant	Validation Accuracy	Test Accuracy*
(f)	1e-5	0.958	0.958 (23/24)
(g)	1e-4	1.000	0.833 (20/24)
(h)	1e-4	0.958	0.958 (23/24)

*(Correct test data/Total test data)

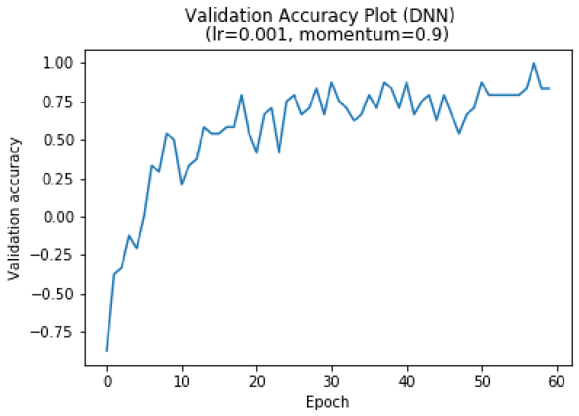
Collision Avoidance

Train Result

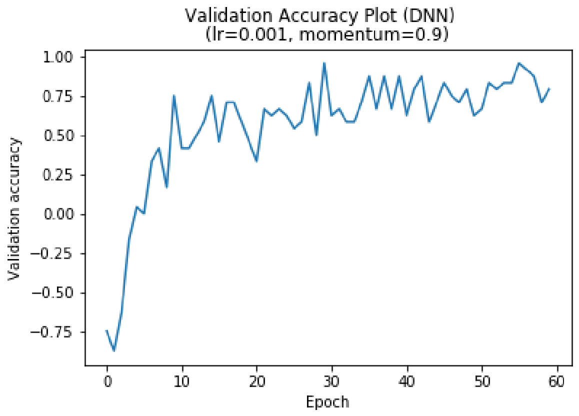
- **Subject**: Effect of the L2 constant



(f)



(g)



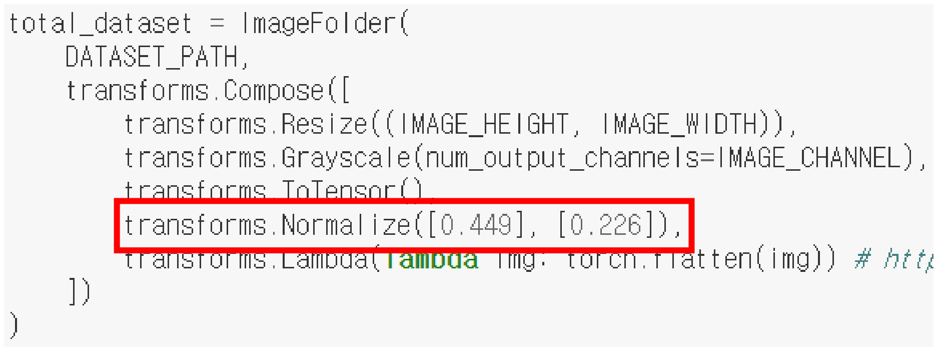
(h)

No Accuracy benefits

Collision Avoidance

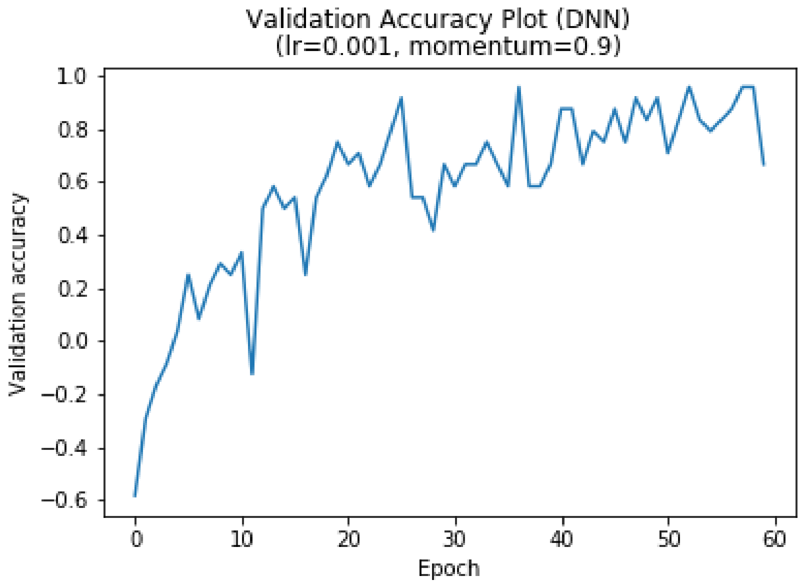
Train Result

- Add normalization at the *transform* part



```
total_dataset = ImageFolder(
    DATASET_PATH,
    transforms.Compose([
        transforms.Resize((IMAGE_HEIGHT, IMAGE_WIDTH)),
        transforms.Grayscale(num_output_channels=IMAGE_CHANNEL),
        transforms.ToTensor(),
        transforms.Normalize([0.449], [0.226]),
        transforms.Lambda(lambda img: torch.flatten(img)) # htt
    ])
)
```

average of the previous 3 elements



Validation accuracy: 0.958
Test accuracy: 0.958 (23/24)

Collision Avoidance

Conclusion

We can do the image classification by using the custom DNN.
However, 95.8% is the maximum accuracy.