**Mathew Varughese**

MAV120

CS 1699

HW 4

**Part I – Implement AlexNet**

AlexNet(

(features): Sequential(

(0): Conv2d(3, 96, kernel\_size=(11, 11), stride=(4, 4))

(1): ReLU(inplace=True)

(2): MaxPool2d(kernel\_size=3, stride=2, padding=0, dilation=1, ceil\_mode=False)

(3): Conv2d(96, 256, kernel\_size=(5, 5), stride=(1, 1), padding=(2, 2))

(4): ReLU(inplace=True)

(5): MaxPool2d(kernel\_size=3, stride=2, padding=0, dilation=1, ceil\_mode=False)

(6): Conv2d(256, 384, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(7): ReLU(inplace=True)

(8): Conv2d(384, 384, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(9): ReLU(inplace=True)

(10): Conv2d(384, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(11): ReLU(inplace=True)

(12): MaxPool2d(kernel\_size=3, stride=2, padding=0, dilation=1, ceil\_mode=False)

)

(flatten): Flatten()

(classifier): Sequential(

(0): Dropout(p=0.5, inplace=False)

(1): Linear(in\_features=9216, out\_features=4096, bias=True)

(2): ReLU(inplace=True)

(3): Dropout(p=0.5, inplace=False)

(4): Linear(in\_features=4096, out\_features=4096, bias=True)

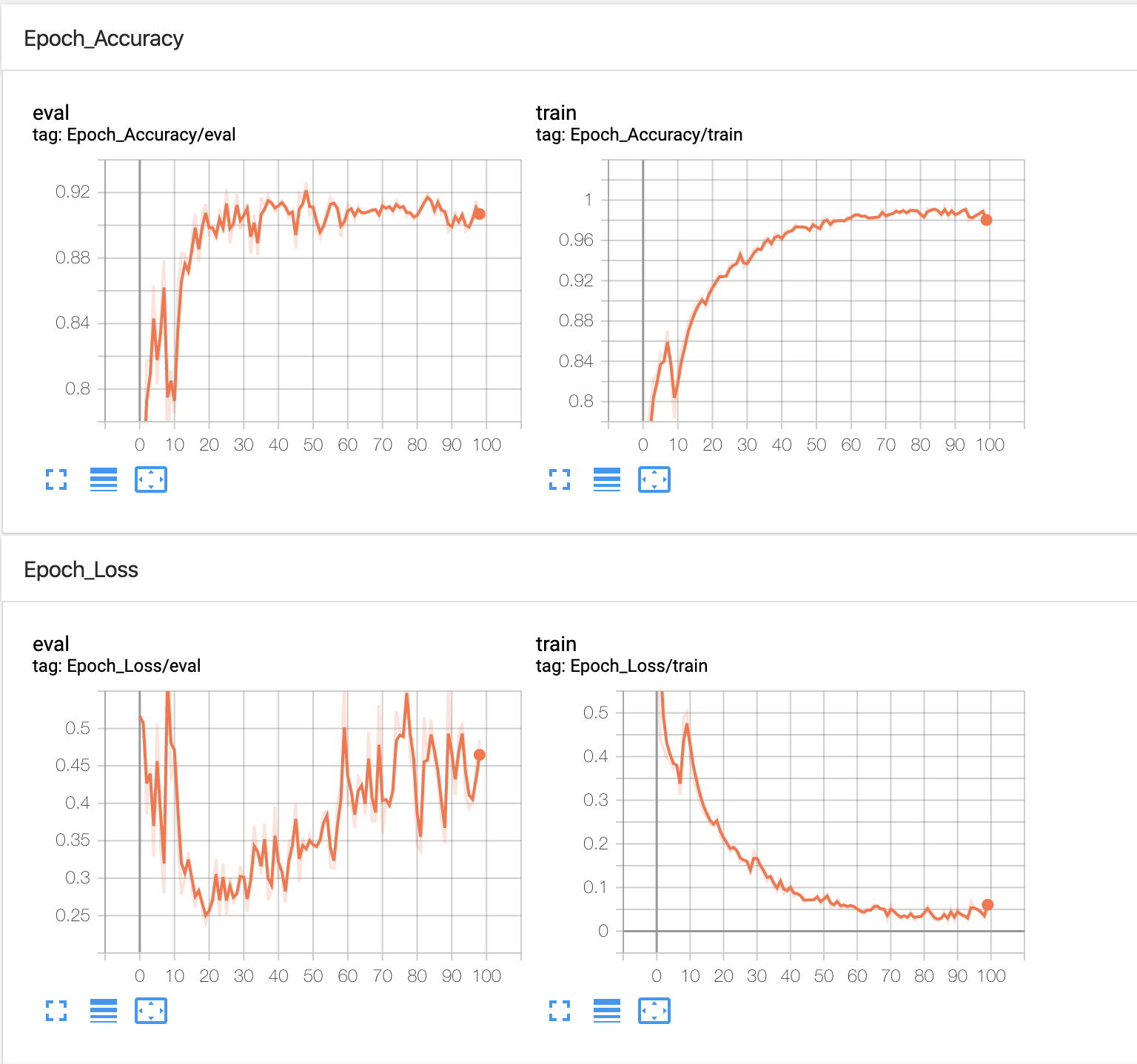
(5): ReLU(inplace=True)

(6): Linear(in\_features=4096, out\_features=4, bias=True)

)

)

**Validation Set Accuracy**: 0.9046



**Part II – Enhancing AlexNet**

**AlexNetLargeKernel**(

(features): Sequential(

(0): Conv2d(3, 96, kernel\_size=(21, 21), stride=(8, 8), padding=(1, 1))

(1): ReLU(inplace=True)

(2): Conv2d(96, 256, kernel\_size=(7, 7), stride=(2, 2), padding=(2, 2))

(3): ReLU(inplace=True)

(4): Conv2d(256, 384, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(5): ReLU(inplace=True)

(6): Conv2d(384, 384, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(7): ReLU(inplace=True)

(8): Conv2d(384, 256, kernel\_size=(3, 3), stride=(2, 2))

(9): ReLU(inplace=True)

)

(flatten): Flatten()

(classifier): Sequential(

(0): Dropout(p=0.5, inplace=False)

(1): Linear(in\_features=9216, out\_features=4096, bias=True)

(2): ReLU(inplace=True)

(3): Dropout(p=0.5, inplace=False)

(4): Linear(in\_features=4096, out\_features=4096, bias=True)

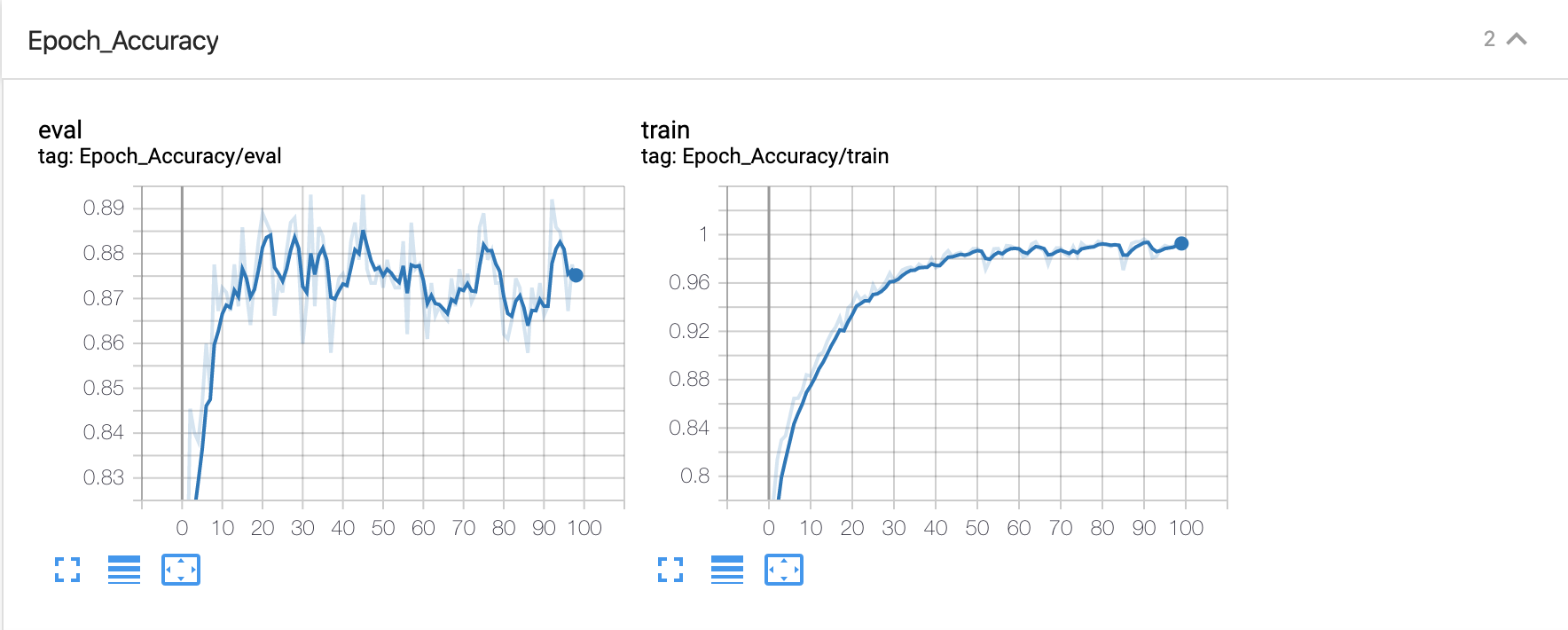
(5): ReLU(inplace=True)

(6): Linear(in\_features=4096, out\_features=4, bias=True)

)

)

**Validation Set Accuracy**: 0.8734



**AlexNetTiny**(

(features): Sequential(

(0): Conv2d(3, 48, kernel\_size=(11, 11), stride=(4, 4))

(1): ReLU(inplace=True)

(2): MaxPool2d(kernel\_size=3, stride=2, padding=0, dilation=1, ceil\_mode=False)

(3): Conv2d(48, 128, kernel\_size=(5, 5), stride=(1, 1), padding=(2, 2))

(4): ReLU(inplace=True)

(5): MaxPool2d(kernel\_size=3, stride=2, padding=0, dilation=1, ceil\_mode=False)

(6): Conv2d(128, 192, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(7): ReLU(inplace=True)

(8): Conv2d(192, 192, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(9): ReLU(inplace=True)

(10): Conv2d(192, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(11): ReLU(inplace=True)

(12): MaxPool2d(kernel\_size=3, stride=2, padding=0, dilation=1, ceil\_mode=False)

)

(flatten): Flatten()

(classifier): Sequential(

(0): Dropout(p=0.5, inplace=False)

(1): Linear(in\_features=4608, out\_features=2048, bias=True)

(2): ReLU(inplace=True)

(3): Dropout(p=0.5, inplace=False)

(4): Linear(in\_features=2048, out\_features=1024, bias=True)

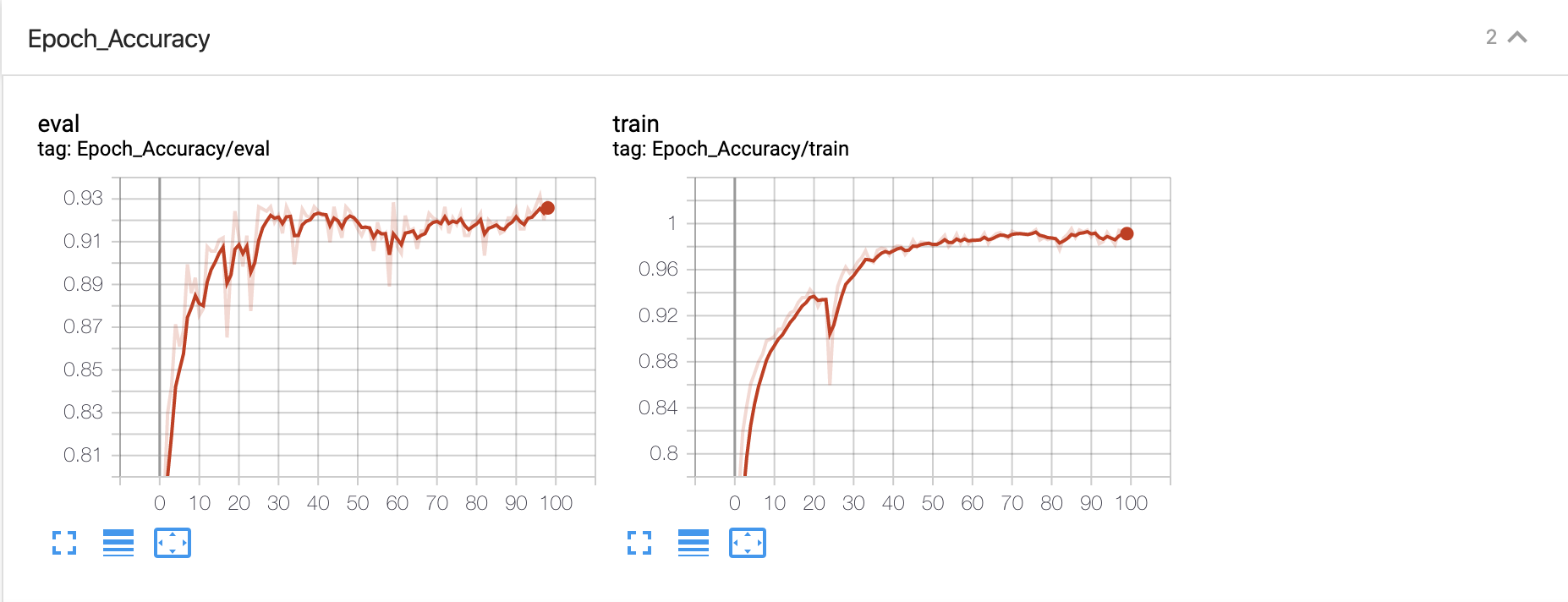
(5): ReLU(inplace=True)

(6): Linear(in\_features=1024, out\_features=4, bias=True)

)

)

**Validation Set Accuracy**: 0.9295



**AlexNetAvgPooling**(

(features): Sequential(

(0): Conv2d(3, 96, kernel\_size=(11, 11), stride=(4, 4))

(1): ReLU(inplace=True)

(2): AvgPool2d(kernel\_size=3, stride=2, padding=0)

(3): Conv2d(96, 256, kernel\_size=(5, 5), stride=(1, 1), padding=(2, 2))

(4): ReLU(inplace=True)

(5): AvgPool2d(kernel\_size=3, stride=2, padding=0)

(6): Conv2d(256, 384, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(7): ReLU(inplace=True)

(8): Conv2d(384, 384, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(9): ReLU(inplace=True)

(10): Conv2d(384, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(11): ReLU(inplace=True)

(12): AvgPool2d(kernel\_size=3, stride=2, padding=0)

)

(flatten): Flatten()

(classifier): Sequential(

(0): Dropout(p=0.5, inplace=False)

(1): Linear(in\_features=9216, out\_features=4096, bias=True)

(2): ReLU(inplace=True)

(3): Dropout(p=0.5, inplace=False)

(4): Linear(in\_features=4096, out\_features=4096, bias=True)

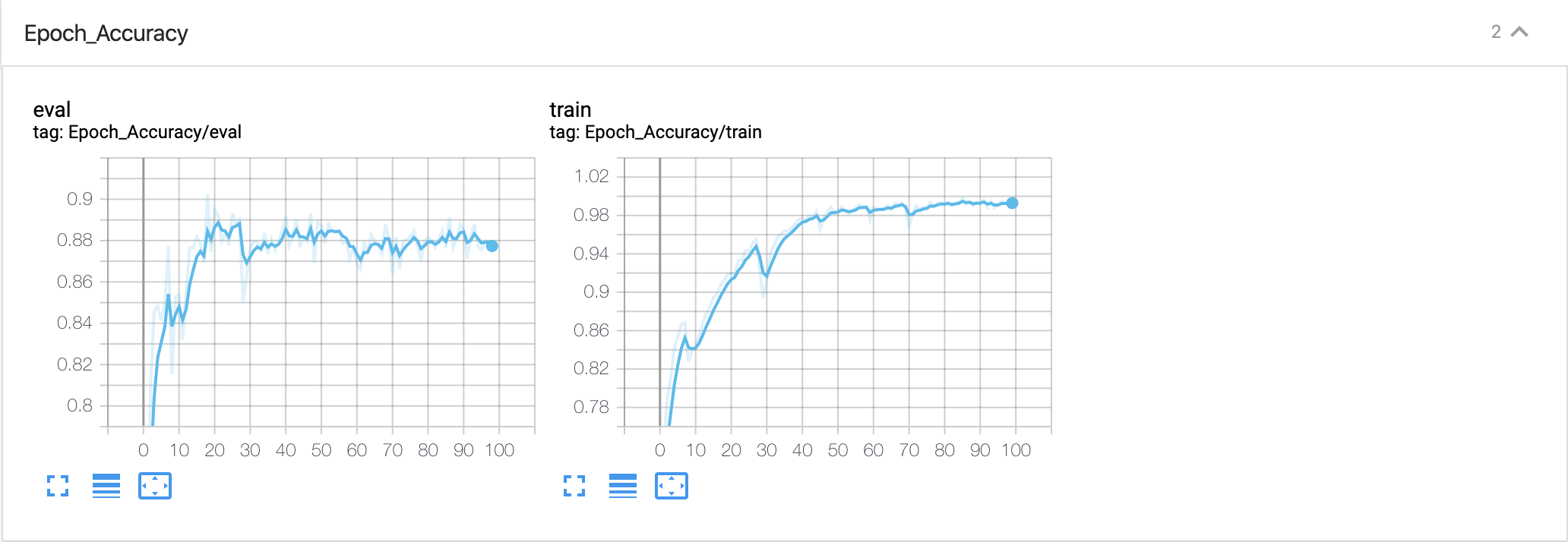
(5): ReLU(inplace=True)

(6): Linear(in\_features=4096, out\_features=4, bias=True)

)

)

**Validation Set Accuracy**: 0.8734



**AlexNetDilation**(

(features): Sequential(

(0): Conv2d(3, 96, kernel\_size=(11, 11), stride=(4, 4), padding=(5, 5), dilation=(2, 2))

(1): ReLU(inplace=True)

(2): MaxPool2d(kernel\_size=3, stride=2, padding=0, dilation=1, ceil\_mode=False)

(3): Conv2d(96, 256, kernel\_size=(5, 5), stride=(1, 1), padding=(4, 4), dilation=(2, 2))

(4): ReLU(inplace=True)

(5): MaxPool2d(kernel\_size=3, stride=2, padding=0, dilation=1, ceil\_mode=False)

(6): Conv2d(256, 384, kernel\_size=(3, 3), stride=(1, 1), padding=(2, 2), dilation=(2, 2))

(7): ReLU(inplace=True)

(8): Conv2d(384, 384, kernel\_size=(3, 3), stride=(1, 1), padding=(2, 2), dilation=(2, 2))

(9): ReLU(inplace=True)

(10): Conv2d(384, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(2, 2), dilation=(2, 2))

(11): ReLU(inplace=True)

(12): MaxPool2d(kernel\_size=3, stride=2, padding=0, dilation=1, ceil\_mode=False)

)

(flatten): Flatten()

(classifier): Sequential(

(0): Dropout(p=0.5, inplace=False)

(1): Linear(in\_features=9216, out\_features=4096, bias=True)

(2): ReLU(inplace=True)

(3): Dropout(p=0.5, inplace=False)

(4): Linear(in\_features=4096, out\_features=4096, bias=True)

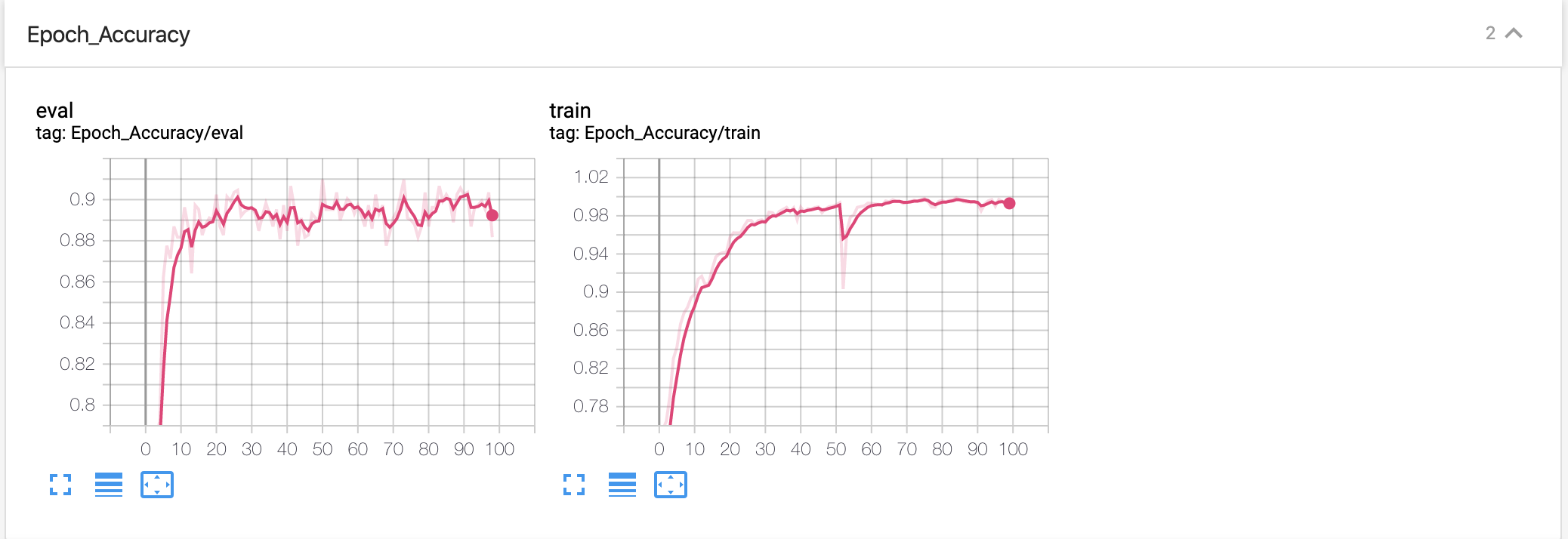
(5): ReLU(inplace=True)

(6): Linear(in\_features=4096, out\_features=4, bias=True)

)

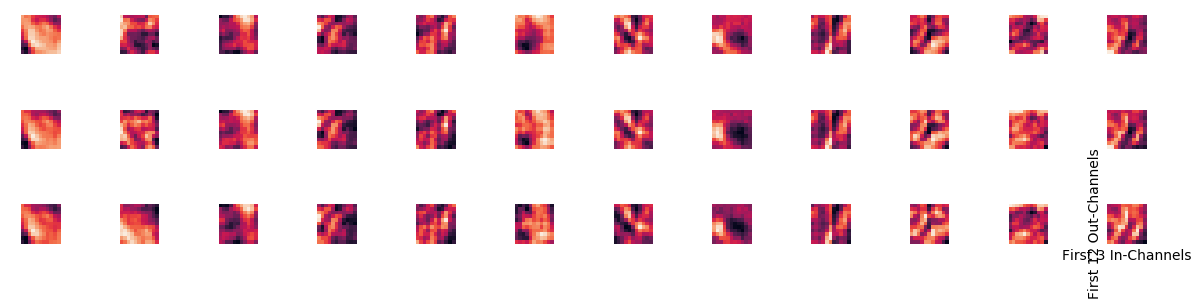
)

**Validation Set Accuracy**: 0.9004



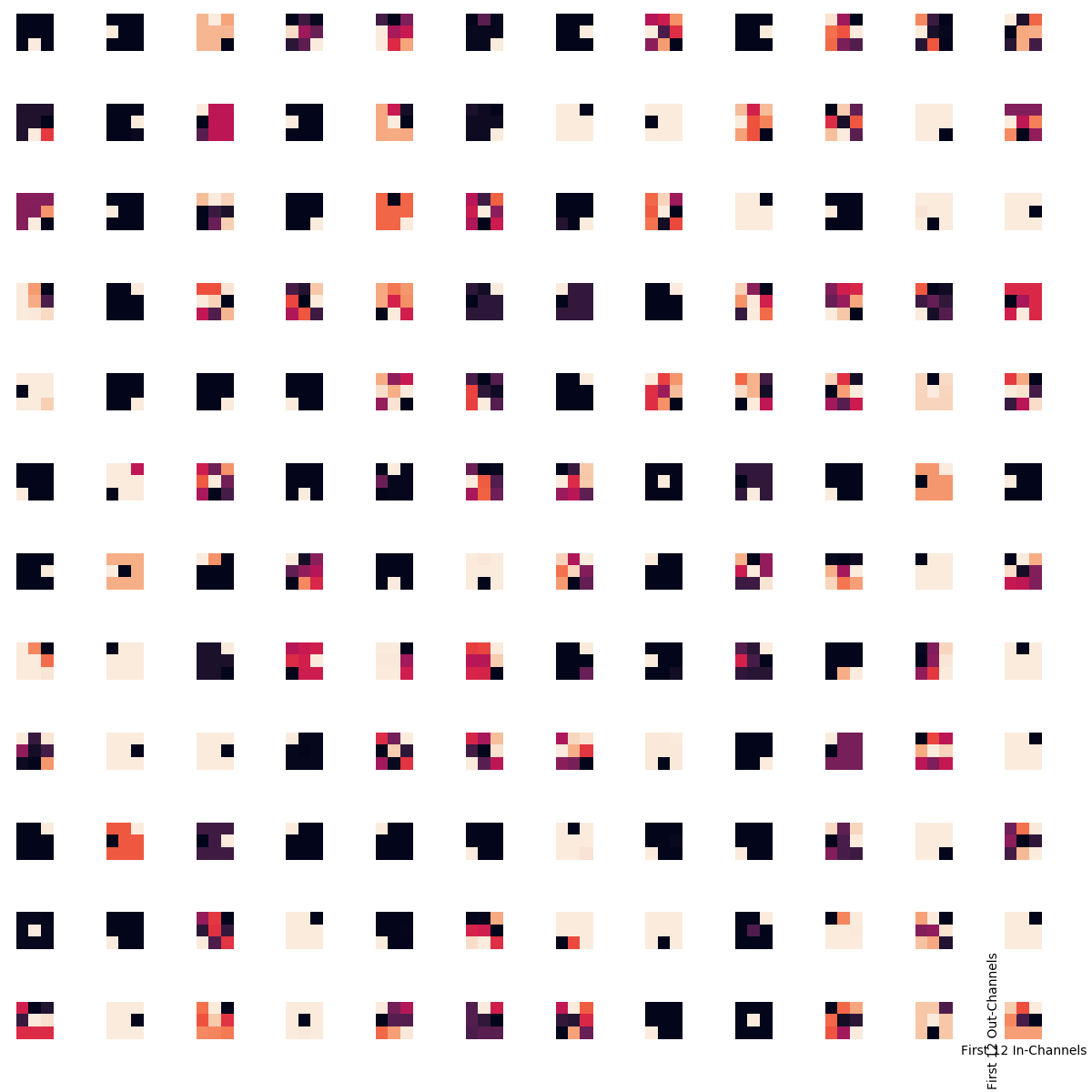
**Part III – Visualizing Learned Filters**

**AlexNet – Domain**

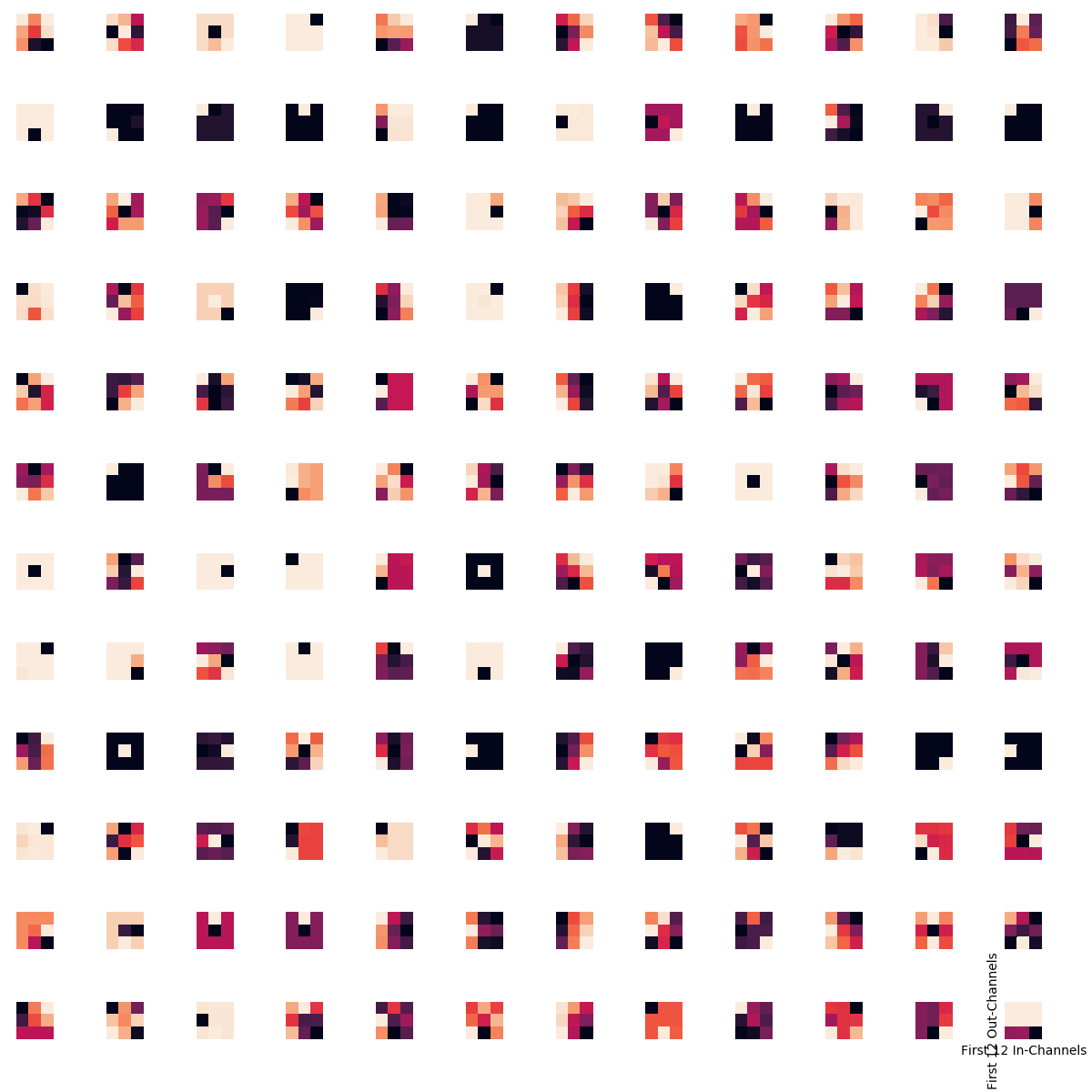
*Conv2d-1*

*****Conv2d-4*

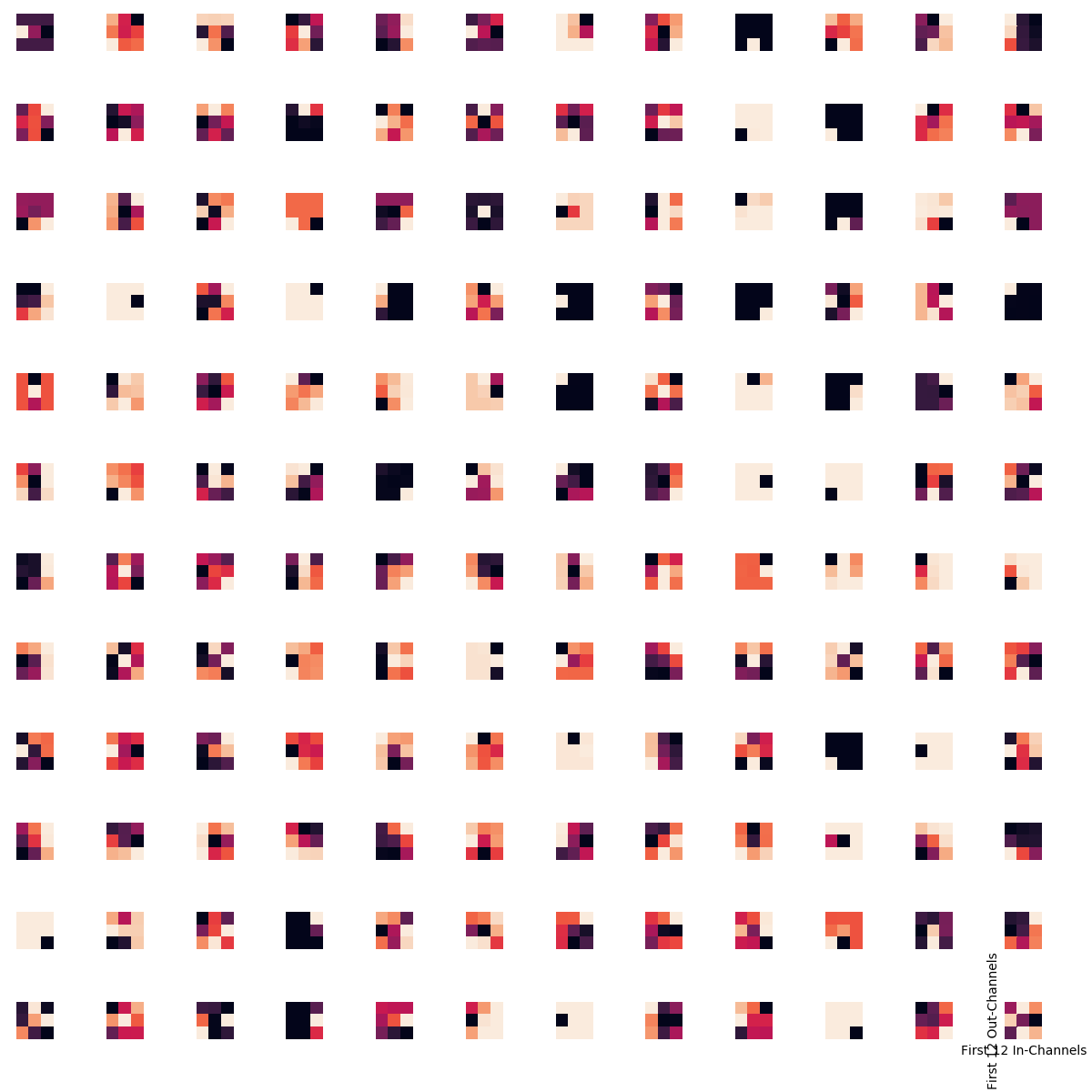
*Conv2d-7*

**

*Conv2d-9*

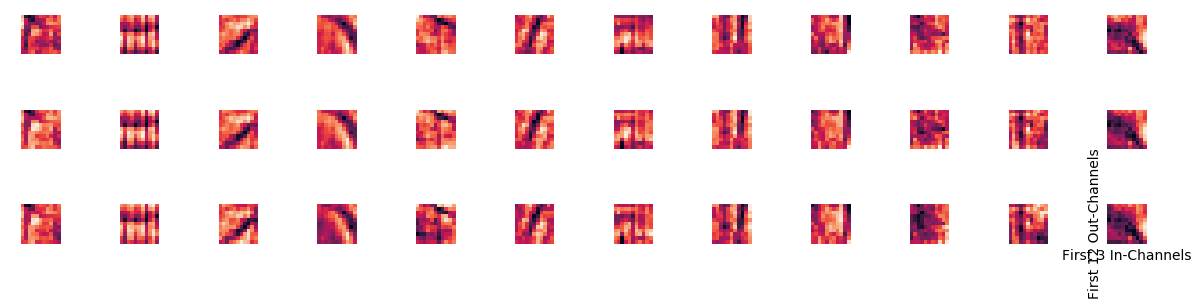
**

*Conv2d-11*

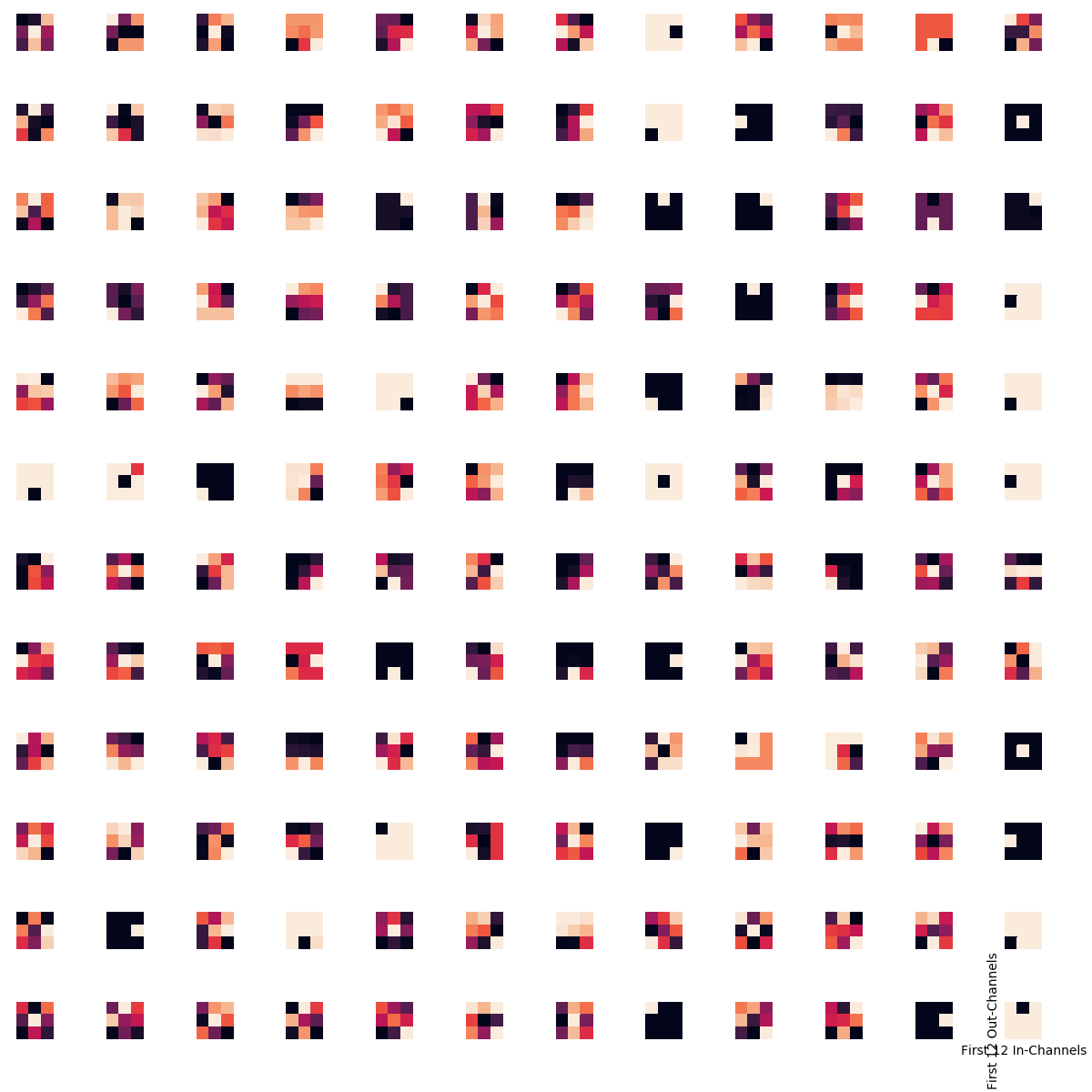
****

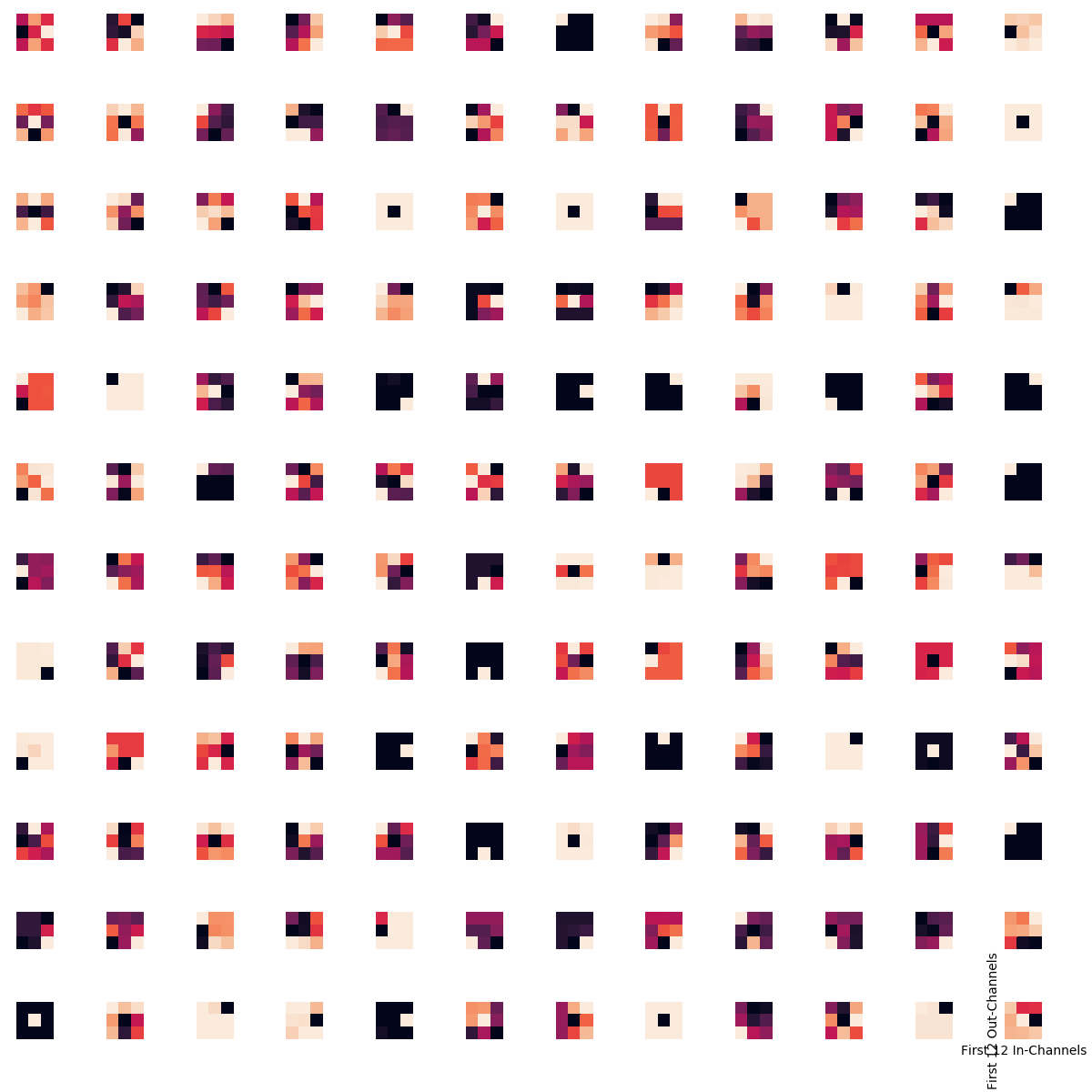
**AlexNet - Class**

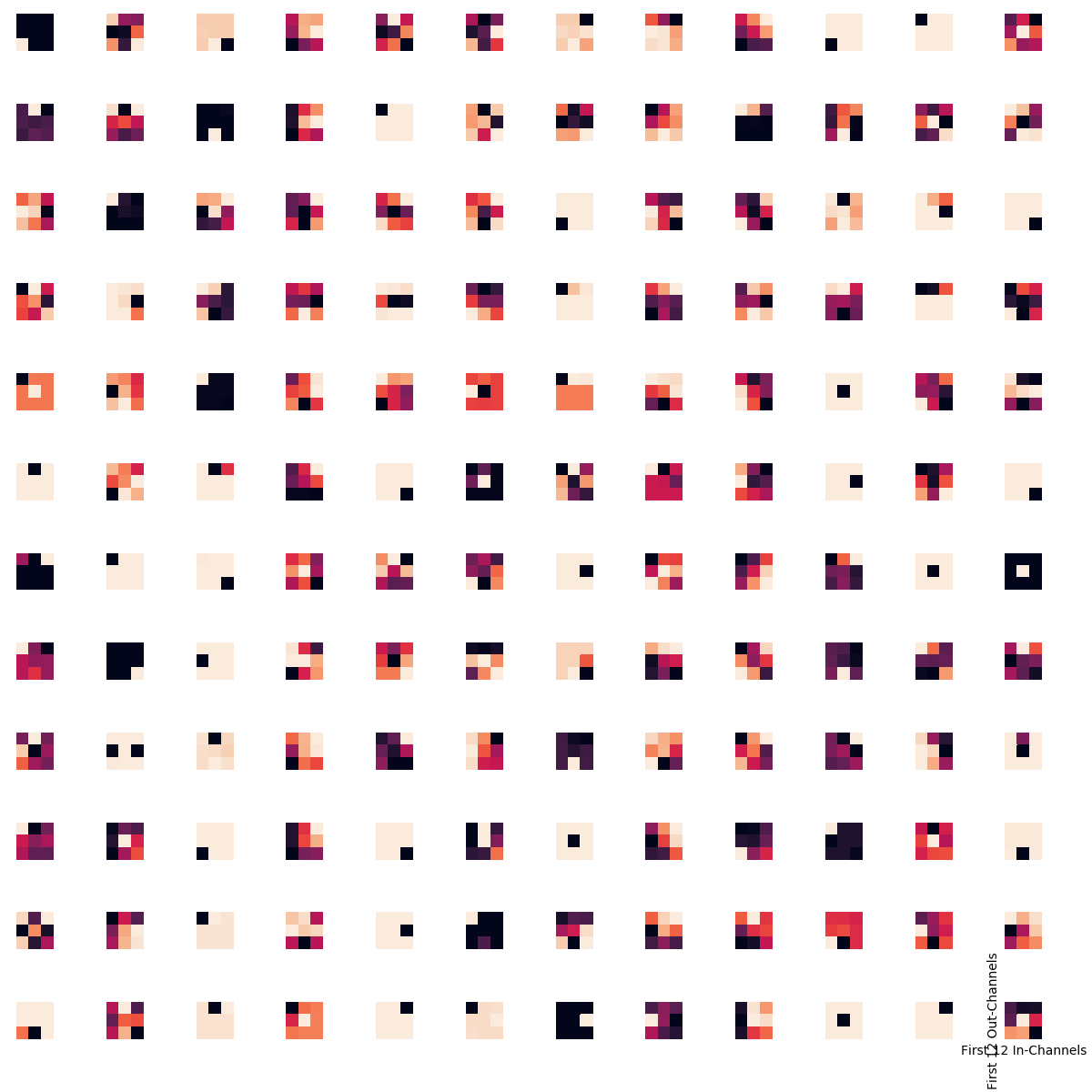
*Conv2d-1*

*Conv2d-4*

**

*Conv2d-7*

*Conv2d-9*

*Conv2d-11*

**Findings**

Domain classification is determining if an image is art, cartoon, photo, or a sketch. It seems that in both the domain and the category label type classification, there were kernels learned that seemed to be pretty standard edge detection and point detection kernels. The Conv2d-11 layer for both category and domain look similar. The domain kernel looks like it is finding some sort of bumps. It is really hard to tell. The category kernel looks like it is finding perhaps textures and different shapes. Maybe this is because it is finding different patterns in different categories.