RAMAIAH
Institute of Technology

Department of Electronics and Communication

Investigation of Rotational Equivariance of Deep Neural Networks

By:

Sinchana S R	1MS17EC106
Soma Rohith	1MS17EC109
Sushmith T	1MS17EC119
Chiranthan K	1MS17EC135

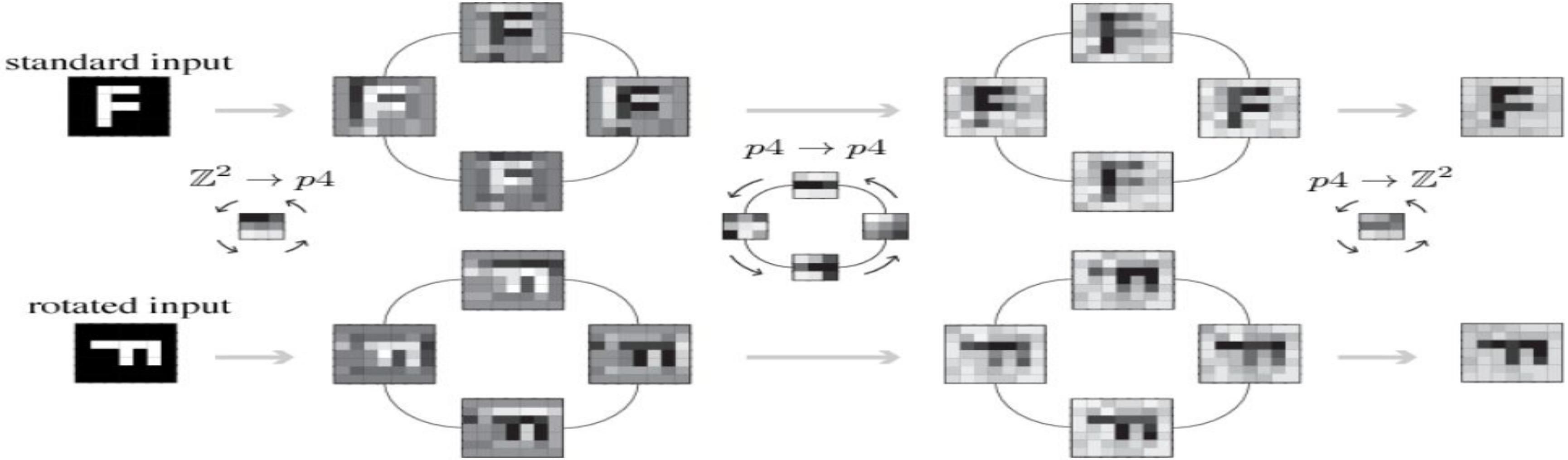
Under the Guidance of
Dr. Raghuram S
Associate Professor
Dept. of E&C,
RIT, Bengaluru

Table of contents

- Introduction
- Literature review
- Methodology
- Results and Discussion
- Progress
- Miscellaneous (Remaining Work)
- References

Introduction

- CNNs are the go-to solution for any task related to image processing,feature learning, etc.
- Feature extraction is independent of spatial position.



Literature Survey

Rotation equivariant CNNs for digital pathology

[1] Veeling, B. S., Linmans, J., Winkens, J., Cohen, T., & Welling, M. (2018, September). Rotation equivariant CNNs for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 210-218). Springer, Cham.
 Cite as: arXiv:1806.03962

Introduction

A new model for digital pathology segmentation is proposed based on the observation that histopathology images are inherently symmetric under rotation and reflection.

Presents analysis which shows improved stability on predictions and demonstrate rotational equivariance which improves tumor detection on challenging lymph node metastases dataset

The model is trained to work on whole-side images(WSI) which is the digitized version of natural images,which exhibit not only translational symmetry but also rotation and reflection symmetry

Generally CNN’s do not exploit these symmetries,so this paper develops a CNN model trained on histopathology data to exhibit fluctuations in predictions under input rotation and reflection

Concepts involved

G-CNNs are a generalization of CNNs that are equivariant under more general symmetry groups which additionally includes reflection.

In a G-CNN, the feature maps are thought of as functions on this group.

In the final layer, a group-pooling layer is used to ensure that the output is either invariant (for classification tasks) or equivariant as a function on the plane (for segmentation tasks, where the output is supposed to transform together with the input)

Model architecture

The architecture is based on densely connected CN, which consist of dense blocks with layers that use the stack of all previous layers as input,alternated with transition blocks consisting of 1x1 convolution layer and 2x2 strided average pooling

The model spatially-pools the input by a factor 2^5 ,as there are 5 dense-block/transition-block pairs

Full model group equivariance is achieved by replacing all convolution layers with group-equivariant versions

The final layer consists of group-pooling layer followed by sigmoid activation

Results and Conclusion (As mentioned in the research paper)

A novel histopathology patch-classification model that outperforms a competitive traditional CNN by enforcing rotation and reflection equivariance is presented.

It demonstrates that rotation equivariance improves reliability of the model

A derived patch-level dataset is presented, allowing straightforward and precise evaluation on a challenging histopathology task.

It has been shown that the rotation equivariance improves reliability of the model, motivating the application and further research of rotation equivariant models in the medical image analysis domain.

Group Equivariant Convolutional Networks

[2] Cohen, Taco, and Max Welling. "Group equivariant convolutional networks." In *International conference on machine learning*, pp. 2990-2999. 2016.C
 “Group equivariant convolutional networks”. (2016)
Cite as: https://arxiv.org/abs/1602.07576

Introduction

G-convolutions increase the expressive capacity of the network without increasing the number of parameters.

Convolutional weight sharing is effective because there is a translation symmetry in most perception tasks: the label function and data distribution are both approximately invariant to shifts. By using the same weights to analyze or model each part of the image, a convolution layer uses far fewer parameters than a fully connected one, while preserving the capacity to learn many useful transformations.

Concepts involved

In this paper they construct representations that have the structure of a linear G-space, for some chosen group G. This means that each vector in the representation space has a pose associated with it, which can be transformed by the elements of some group of transformations G. This additional structure allows to model data more efficiently: A filter in a G-CNN detects co-occurrences of features that have the preferred relative pose, and can match such a feature constellation in every global pose through an operation called the G-convolution.

Results and Conclusion (As mentioned in the research paper)

By exploiting symmetries, G-CNN's achieve state of the art results on rotated MNIST and CIFAR10. The general theory of G-CNNs for discrete groups, showing that all layer types are equivariant to the action of the chosen group G has been developed.

The experimental results show that G-convolutions can be used as a drop-in replacement for spatial convolutions in modern network architectures, improving their performance without further tuning.

Roto-Translation Covariant Convolutional Networks for Medical Image Analysis

[3] Bekkers, E.J., Lafarge, M.W., Veta, M., Eppenhof, K.A., Pluim, J.P. and Duits, R., 2018, September. Roto-translation covariant convolutional networks for medical image analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 440-448). Springer, Cham.
Cite as: https://arxiv.org/abs/1804.03393

Introduction

The convolution layers are defined in terms of representations of the special Euclidean motion group SE(2). The framework for rotation and translation covariant deep learning using SE(2) group convolutions has been proposed in the paper. The group product of this special Euclidean motion group SE(2) describes how a concatenation of two roto-translations results in a net roto-translation.

Concepts involved

A lifting layer is introduced which lifts a 2D vector valued image to an SE(2)-image which is the 3D vector valued data whose domain is SE(2)

Further, a group convolution layer from and to an SE(2)-image is substantially created

Additionally, a projection layer from an SE(2)-image to a 2D image is also created

The lifting and group convolution layers are SE(2) covariant making the output roto-translates with the input.

In the final projection layer, a maximum intensity projection over rotations, makes the full CNN rotation invariant.

Results and Conclusion (As mentioned in the research paper)

A consistent improvement of performances was observed across the three medical image analysis tasks when using G-CNNs compared to their corresponding CNN baselines.

The reported results are in line with the benchmark of each dataset and the best performances were obtained for an orientation capacity $N \geq 4$, indicating the advantage of learning such rotation-invariant representations

It has been concluded that it is beneficiary to include SE(2) group convolution layers in CNN network design, as this avoids the need for rotation augmentation and it improves overall performance

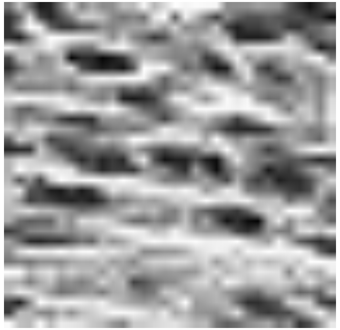
Methodology

- Group Convolution Layer can be viewed as a process in which we first transform/rotate the filter and then compute the inner product.
- This allows the network to learn feature maps associated with different rotated versions of the input image in a single pass.
- We are comparing results of SE2CNN group convolution and a standard CNN using a model like resnet50 or alexnet.Comparing results of both to predict cancer, inturn investigating rotational equivariance of deep neural networks.

Results
Image Processing

96*96 pixels
RGB
.tif format
27-30 kB size

Central 32*32 pixels
RGB
.jpeg format
1-3 kB size



Central 32*32 pixels
Gray
.jpeg format
< 2 kB size

SE2CNN Results

```
-----
RAW kernel shapes:
[Layer_1/kernel:0]: (5, 5, 1, 4), total nr of weights = 100
[Layer_2/kernel:0]: (5, 5, 12, 4, 8), total nr of weights = 9600
[Layer_3/kernel:0]: (5, 5, 96, 16), total nr of weights = 38400
[Layer_4/kernel:0]: (1, 1, 16, 128), total nr of weights = 2048
[Layer_5/kernel:0]: (1, 1, 128, 2), total nr of weights = 256
```

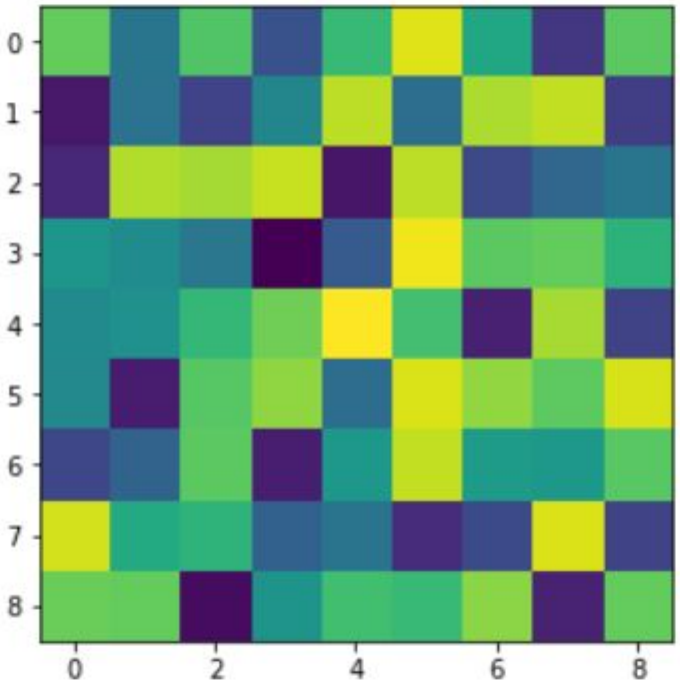
```
 Length of train_data
176020
 Length of eval_data
44005
 Length of train_labels
176020
 Length of eval_labels
44005
 inputs_ph
Tensor("Placeholder:0", shape=(?, 36, 36, 1), dtype=float32)
 labels_ph
Tensor("Placeholder_1:0", shape=(?,), dtype=int32)
```

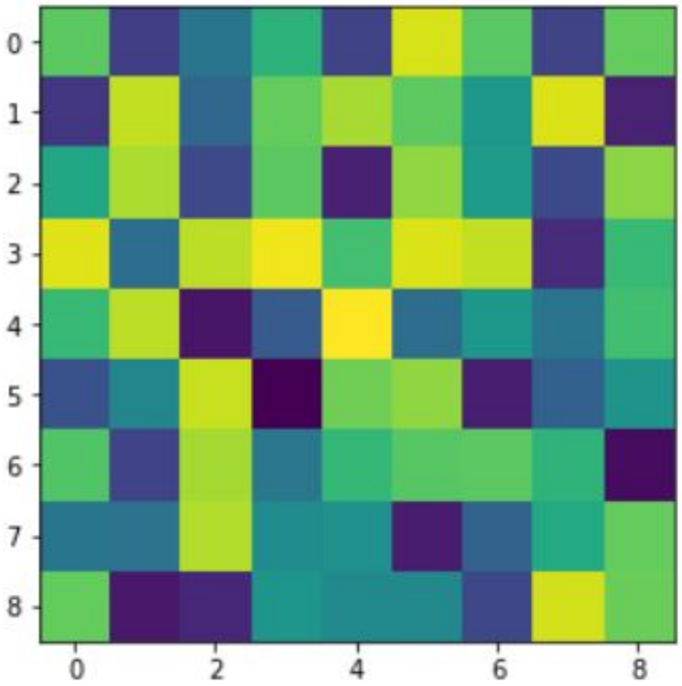
```
Z2-SE2N BASE KERNEL SHAPE: (5, 5, 1, 4)
WARNING:tensorflow:From /home/sinchu-sr/Mini-Project/SE2CNN/demo/../se2cnn/laye
rs.py:251: The name tf.sparse_tensor_dense_matmul is deprecated. Please use tf.
sparse.sparse_dense_matmul instead.

Z2-SE2N ROTATED KERNEL SET SHAPE: (12, 5, 5, 1, 4)
OUTPUT SE2N ACTIVATIONS SHAPE: (?, 32, 32, 12, 4)
WARNING:tensorflow:From /home/sinchu-sr/Mini-Project/SE2CNN/demo/../se2cnn/laye
rs.py:191: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d i
nstead.

(?, 16, 16, 12, 4)
SE2N-SE2N BASE KERNEL SHAPE: (5, 5, 12, 4, 8)
SE2N-SE2N ROTATED KERNEL SET SHAPE: (12, 5, 5, 12, 4, 8)
OUTPUT SE2N ACTIVATIONS SHAPE: (?, 12, 12, 12, 8)
(?, 6, 6, 12, 8)
(?, 2, 2, 16)
(?, 2, 2, 128)
(?, 2)
```

```
Epoch  0  finished... Average loss =  1.2581 , time =  1121.4208
Epoch  1  finished... Average loss =  0.6755 , time =  1106.8087
Epoch  2  finished... Average loss =  0.6755 , time =  1106.3753
Epoch  3  finished... Average loss =  0.6755 , time =  1349.9832
Epoch  4  finished... Average loss =  0.6754 , time =  2178.2853
Epoch  5  finished... Average loss =  0.6755 , time =  2187.3592
Epoch  6  finished... Average loss =  0.6755 , time =  2205.9988
Epoch  7  finished... Average loss =  0.6754 , time =  2191.4852
Epoch  8  finished... Average loss =  0.6754 , time =  1456.6467
Epoch  9  finished... Average loss =  0.6754 , time =  1172.833
 Compare the first 50 results with the ground truth
 The first 50 predicted results
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0]
 The first 50 ground truth (actual results)
[1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0.
 1. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0.
 1. 1.]
 The accuracy (average number of successes)
0.5982501988410408
 Total number of errors
17679.0
 Error rate
40.17498011589592
Confusion matrix, without normalization
sinchu-sr@ubuntu:~/Mini-Project/SE2CNN/demo$
```

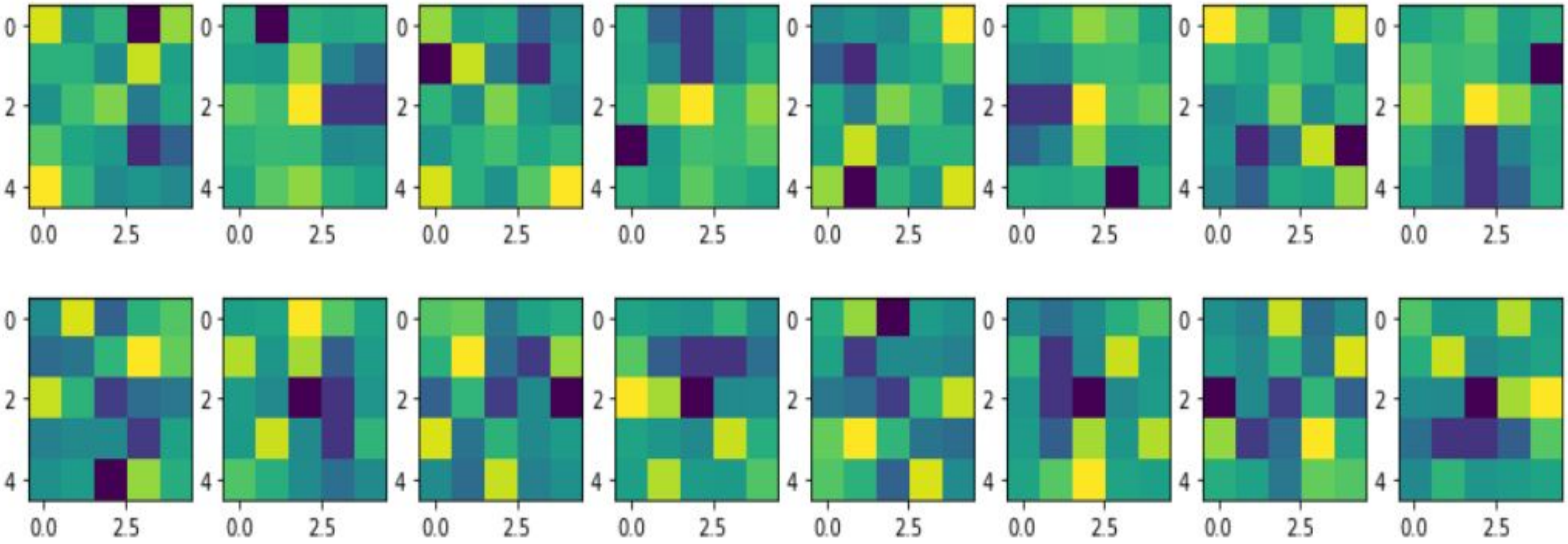
SE2CNN Covariance Test Results

Random 2D Input Signal and its 90 degree rotated version



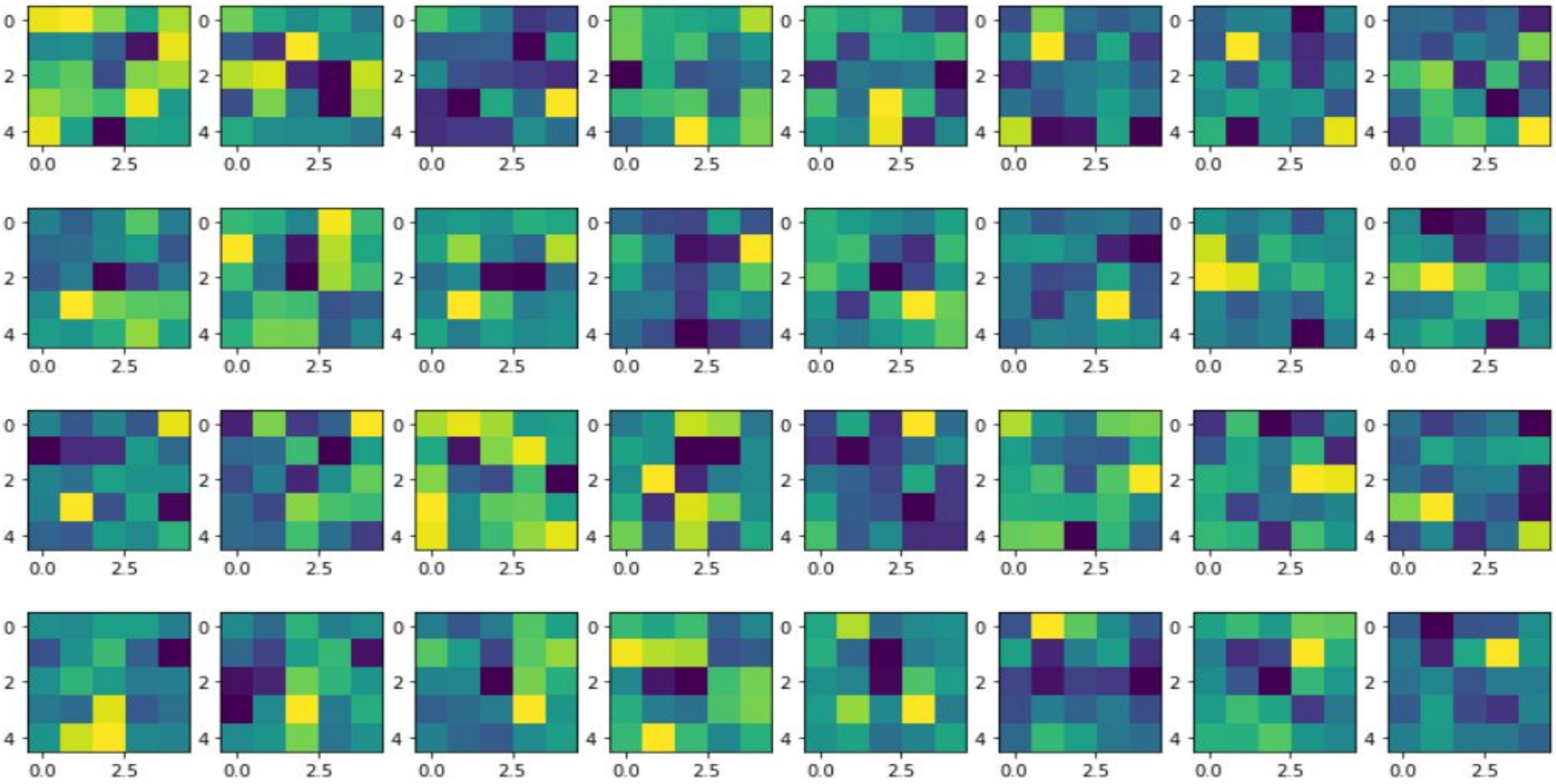


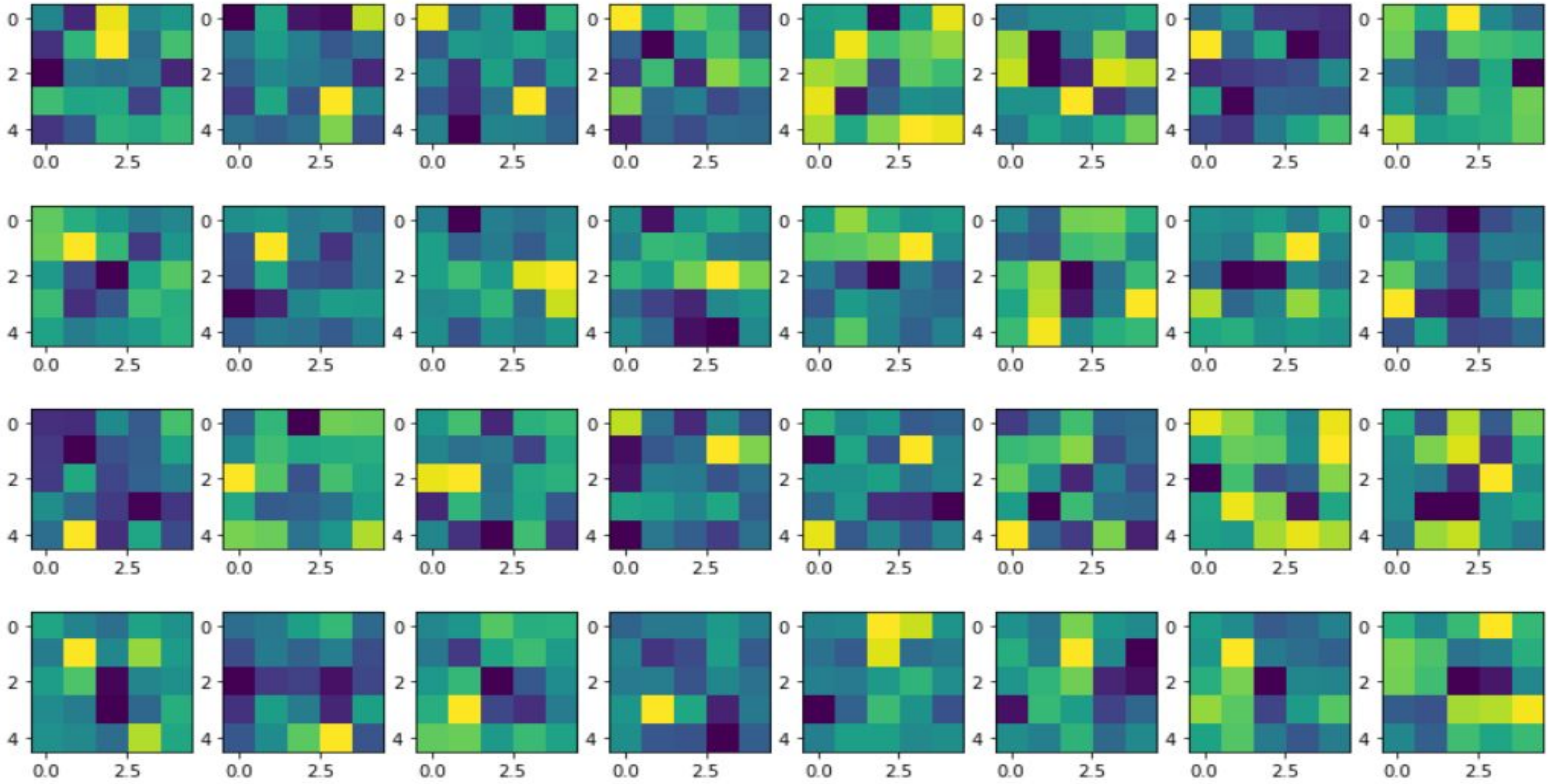
The rotated set of a lifting kernel



Rotated versions of the lifting kernels

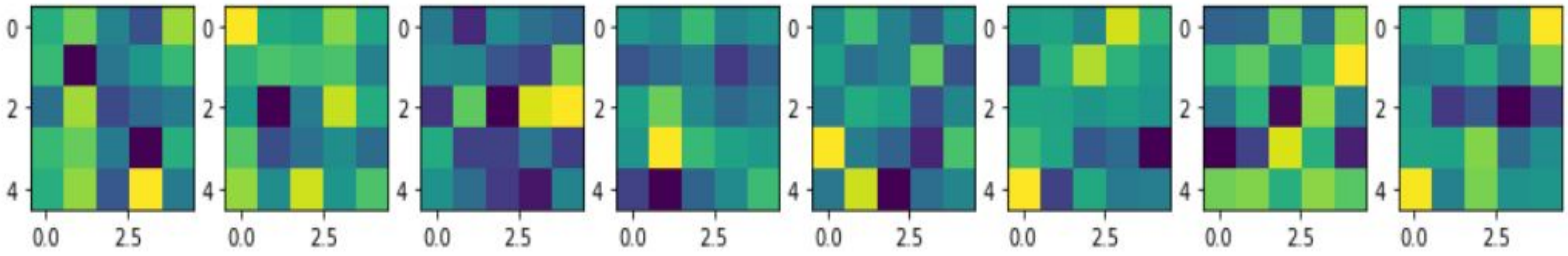
The rotate gconv kernels



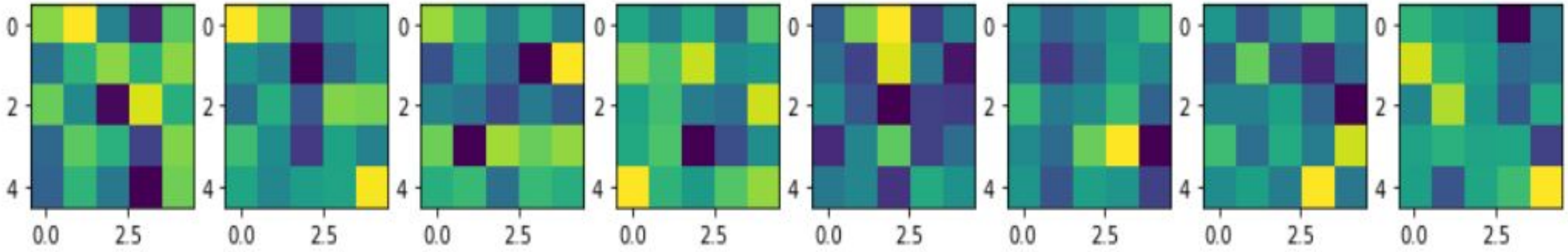


Each row is a rotated version of an se2 kernel (planar rotation + shift in theta direction)

Check covariance of the lifting layer

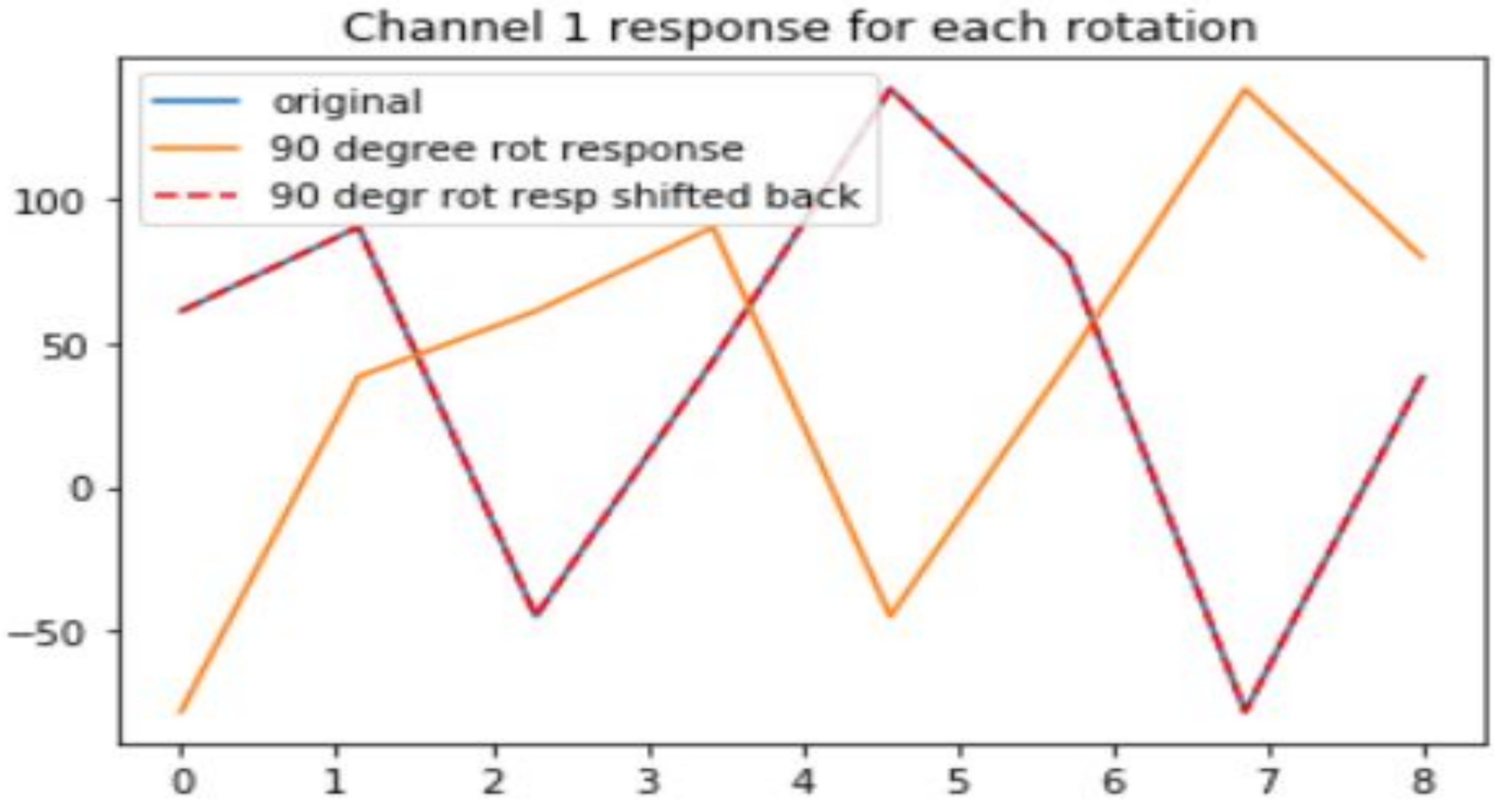


The unrotated input.



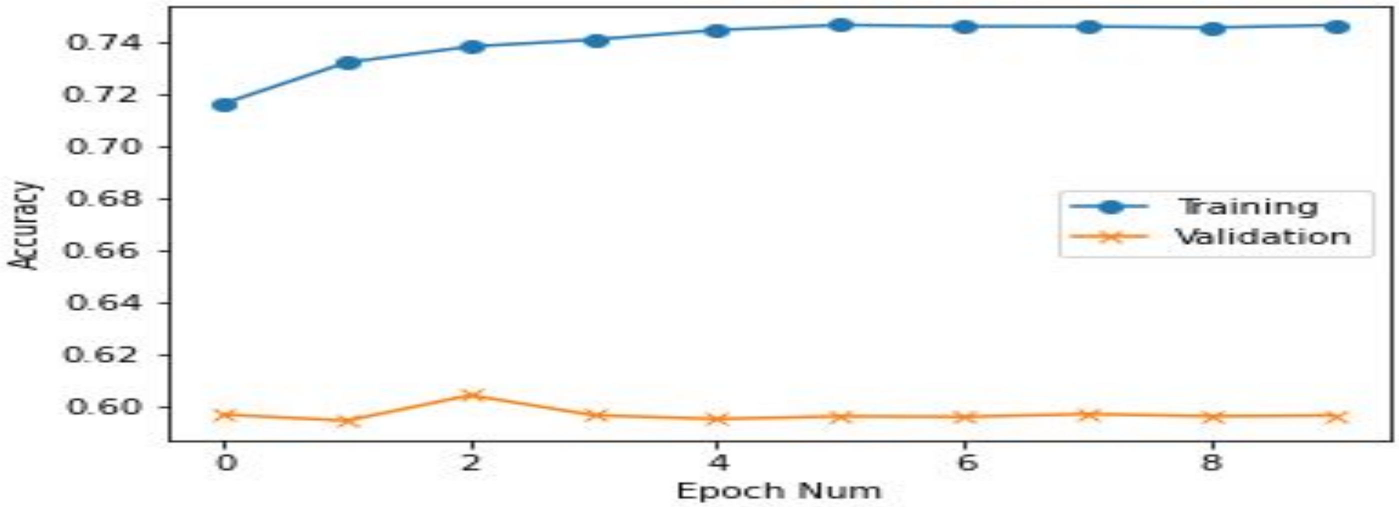
The 90-degree rotated input. Notice a shift of Ntheta/4 (=90/360*Ntheta) bins and a rotation of 90 of each plane.

Check covariance down to the output of the group convolution layer

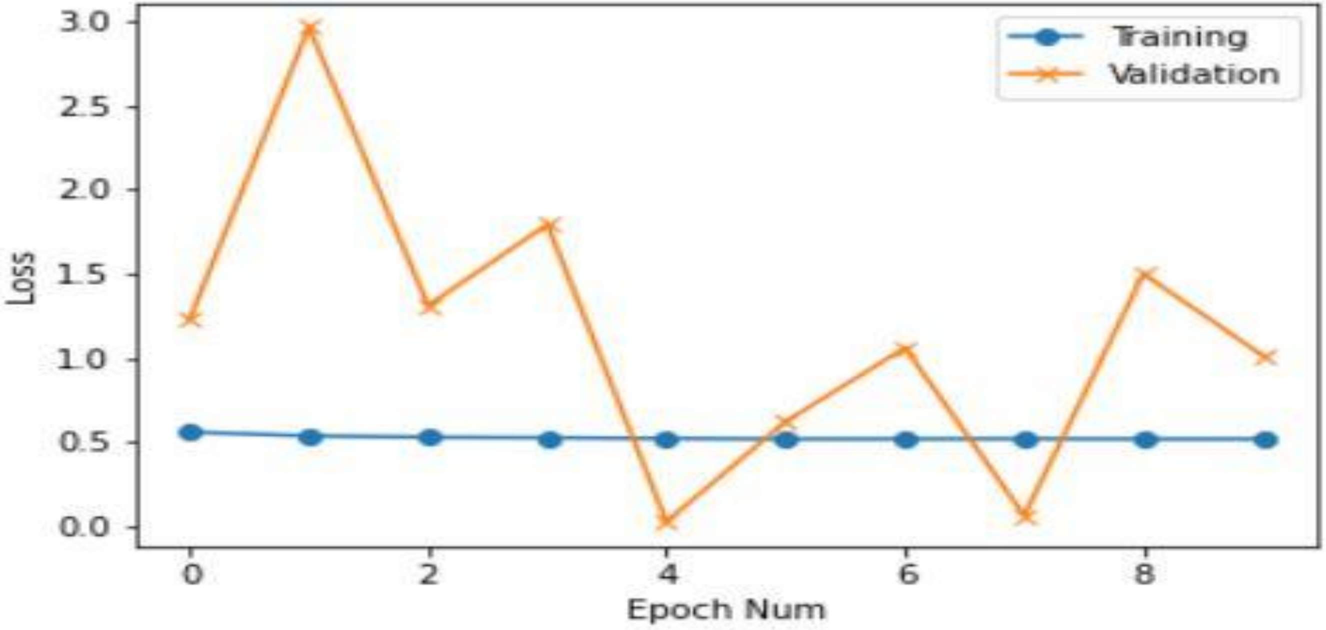


Results of Resnet50 model Using MONK library with Keras Backend

Train v/s Validation Accuracy Curve



Train v/s Validation Loss Curve



```
Dataset Details
    Train path:     /kaggle/input/cancer/train
    Val path:       None
    CSV train path: None
    CSV val path:   None
    Label Type:     single

Dataset Params
    Input Size:   224
    Batch Size:   4
    Data Shuffle: True
    Processors:   2
    Train-val split:   0.7

Found 123215 images belonging to 2 classes.
Found 52805 images belonging to 2 classes.
Pre-Composed Train Transforms
[{'RandomHorizontalFlip': {'p': 0.8}}, {'MeanSubtraction': {'mean': [0.485, 0.456, 0.406]}}, {'RandomHorizontalFlip': {'p': 0.8}}, {'MeanSubtract
ion': {'mean': [0.485, 0.456, 0.406]}}, {'RandomHorizontalFlip': {'p': 0.8}}, {'MeanSubtraction': {'mean': [0.485, 0.456, 0.406]}}]

Pre-Composed Val Transforms
[{'RandomHorizontalFlip': {'p': 0.8}}, {'MeanSubtraction': {'mean': [0.485, 0.456, 0.406]}}, {'RandomHorizontalFlip': {'p': 0.8}}, {'MeanSubtract
ion': {'mean': [0.485, 0.456, 0.406]}}, {'RandomHorizontalFlip': {'p': 0.8}}, {'MeanSubtraction': {'mean': [0.485, 0.456, 0.406]}}]
```

```
Dataset Numbers
    Num train images: 123215
    Num val images:   52805
    Num classes:      2

Model Params
    Model name:          resnet50
    Use Gpu:             True
    Gpu Memory Fraction: 0.6
    Use pretrained:      True
    Freeze base network: True

Model Details
    Loading pretrained model
    Model Loaded on device
        Model name:                          resnet50
        Num layers in model:  108
        Num trainable layers: 2

Optimizer
    Name:          sgd
    Learning rate: 0.0001
    Params:        {'lr': 0.0001, 'momentum': 0.9, 'weight_decay': 0, 'momentum_dampening_rate': 0, 'clipnorm': 0.0, 'clipvalue': 0.0}
```

```
Learning rate scheduler
    Name:   reduceonplateaulr
    Params: {'mode': 'min', 'factor': 0.1, 'patience': 3, 'verbose': True, 'threshold': 0.0001, 'threshold_mode': 'rel', 'cooldown': 0, 'min_lr': 0, 'epsilon': 1e-08}

Loss
    Name:          crossentropy
    Params:        {'weight': None, 'batch_axis': 0, 'axis_to_sum_over': -1, 'label_as_categories': True, 'label_smoothing': False}

Training params
    Num Epochs: 10

Display params
    Display progress:           True
    Display progress realtime: True
    Save Training logs:         True
    Save Intermediate models:   True
    Intermediate model prefix: intermediate_model_
```

Rectangular Snip

```
Training Start
Epoch 1/10
30803/30803 [==============================] - 1153s 37ms/step - loss: 0.5588 - accuracy: 0.7165 - val_loss: 1.2297 - val_accuracy: 0.5972

Epoch 00001: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/resume_state.h5

Epoch 00001: val_loss improved from inf to 1.22971, saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/best_model.h5

Epoch 00001: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/intermediate_model_01.h5
Epoch 2/10
30803/30803 [==============================] - 1199s 39ms/step - loss: 0.5363 - accuracy: 0.7323 - val_loss: 2.9629 - val_accuracy: 0.5948

Epoch 00002: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/resume_state.h5

Epoch 00002: val_loss did not improve from 1.22971
```

```
Epoch 00002: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/intermediate_model_02.h5
Epoch 3/10
30803/30803 [==============================] - 1159s 38ms/step - loss: 0.5280 - accuracy: 0.7384 - val_loss: 1.3103 - val_accuracy: 0.6046

Epoch 00003: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/resume_state.h5

Epoch 00003: val_loss did not improve from 1.22971

Epoch 00003: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/intermediate_model_03.h5
Epoch 4/10
30803/30803 [==============================] - 1167s 38ms/step - loss: 0.5253 - accuracy: 0.7410 - val_loss: 1.7915 - val_accuracy: 0.5969

Epoch 00004: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/resume_state.h5

Epoch 00004: val_loss did not improve from 1.22971
Epoch 00004: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/intermediate_model_04.h5

Epoch 00004: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-06.
Epoch 5/10
30803/30803 [==============================] - 1170s 38ms/step - loss: 0.5207 - accuracy: 0.7447 - val_loss: 0.0267 - val_accuracy: 0.5955

Epoch 00005: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/resume_state.h5

Epoch 00005: val_loss improved from 1.22971 to 0.02670, saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/
best_model.h5

Epoch 00005: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/intermediate_model_05.h5
Epoch 6/10
30803/30803 [==============================] - 1168s 38ms/step - loss: 0.5178 - accuracy: 0.7467 - val_loss: 0.6220 - val_accuracy: 0.5965

Epoch 00006: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/resume_state.h5
```

```
Epoch 00006: val_loss did not improve from 0.02670

Epoch 00006: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/intermediate_model_06.h5
Epoch 7/10
30803/30803 [==============================] - 1153s 37ms/step - loss: 0.5179 - accuracy: 0.7461 - val_loss: 1.0594 - val_accuracy: 0.5963

Epoch 00007: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/resume_state.h5

Epoch 00007: val_loss did not improve from 0.02670

Epoch 00007: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/intermediate_model_07.h5
Epoch 8/10
30803/30803 [==============================] - 1161s 38ms/step - loss: 0.5190 - accuracy: 0.7461 - val_loss: 0.0634 - val_accuracy: 0.5974

Epoch 00008: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/resume_state.h5

Epoch 00008: val_loss did not improve from 0.02670

Epoch 00008: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/intermediate_model_08.h5

Epoch 00008: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-07.
Epoch 9/10
30803/30803 [==============================] - 1165s 38ms/step - loss: 0.5175 - accuracy: 0.7457 - val_loss: 1.4982 - val_accuracy: 0.5965

Epoch 00009: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/resume_state.h5

Epoch 00009: val_loss did not improve from 0.02670

Epoch 00009: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/intermediate_model_09.h5
Epoch 10/10
30803/30803 [==============================] - 1176s 38ms/step - loss: 0.5173 - accuracy: 0.7466 - val_loss: 1.0139 - val_accuracy: 0.5968
```

```
Epoch 00010: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/resume_state.h5

Epoch 00010: val_loss did not improve from 0.02670

Epoch 00010: saving model to workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/intermediate_model_10.h5
    Training completed in: 194m 32s
    Best val Acc:          0.604591

Training End

Training Outputs
    Model Dir:   /kaggle/working/workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/models/
    Log Dir:     /kaggle/working/workspace/Cancer-Detection-Using-MONK/Using-Keras-Backend/output/logs/
    Final model: final
    Best model:  best_model
    Log 1 - Validation accuracy history log: val_acc_history.npy
    Log 2 - Validation loss history log:     val_loss_history.npy
    Log 3 - Training accuracy history log:   train_acc_history.npy
    Log 4 - Training loss history log:       train_loss_history.npy
    Log 5 - Training curve:                  train_loss_history.npy
    Log 6 - Validation curve:                train_loss_history.npy
```

```
<Figure size 432x288 with 0 Axes>
```

```
Dataset Details
    Test path:     /kaggle/input/cancer/validation
    CSV test path:  None

Dataset Params
    Input Size:  224
    Processors:   2

Found 44005 images belonging to 2 classes.
Pre-Composed Test Transforms
[{'MeanSubtraction': {'mean': [0.485, 0.456, 0.406]}}, {'MeanSubtraction': {'mean': [0.485, 0.456, 0.406]}}, {'MeanSubtraction': {'mean': [0.485, 0.456, 0.406]}}]

Dataset Numbers
    Num test images: 44005
    Num classes:      2
```

Rectangular Snip

```
Testing
44005/44005 [==============================] - 858s 20ms/step

    Result
        class based accuracies
            0. CANCER - 0.9955314214604898 %
            1. Normal - 99.92402947656309 %
        total images:               44005
        num correct predictions: 26482
        Average accuracy (%):     60.17952505397114
```

Inferences on test images

[21]:

```
img_name = "/kaggle/input/cancer/test/c2 (10014).jpeg";
predictions = gtf.Infer(img_name=img_name);

#Display
from IPython.display import Image
Image(filename=img_name)
```

```
Prediction
    Image name:          /kaggle/input/cancer/test/c2 (10014).jpeg
    Predicted class:       Normal
    Predicted score:       0.9907813668251038
```

Out[21]:

Discussion

- SE2CNN model (group convolution) having only 5 layers gave an accuracy of 59.825%
- Meanwhile a standard CNN using an model such as RESNET50 even though having 108 layers in model with 2 trainable layer gave an average accuracy of 60.1795%
- So if we add few more layers to SE2CNN it might perform better than standard CNN as it considers rotational equivariance property.

Progress

- We have processed the image and modified the dataset.
    - The original DATASET is https://www.kaggle.com/c/histopathologic-cancer-detection/data
    - The modified dataset contains images of original dataset which is renamed , cropped to central 32*32 region , converted to gray and .jpeg format.It is cropped because in the original dataset a positive label indicates that the center 32x32px region of a patch contains at least one pixel of tumor tissue.
    - 20% of train images of original dataset is in validation folder 80% of train images of original dataset is in train folder All images of test folder of original dataset is in test folder

- Both train and validation folder of this modified dataset has CANCER and Normal folder images are put to this CANCER and Normal folder based on the label provided in the train_labels.csv of the original dataset.
- Standard CNN uses this modified dataset.
- SE2CNN uses the original dataset but the images are cropped to central 32*32 region , converted to gray and .jpeg format.

- We have trained our dataset on various models and have increased accuracy from 40% to 60%

Miscellaneous (Remaining Work)

- Train on different models ,increase the number of epochs and change hyper-parameters and train again to increase accuracy.

References

[1] Veeling, B. S., Linmans, J., Winkens, J., Cohen, T., & Welling, M. (2018, September). Rotation equivariant CNNs for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 210-218). Springer, Cham.
Cite as: arXiv:1806.03962

[2] Cohen, Taco, and Max Welling. "Group equivariant convolutional networks." In *International conference on machine learning*, pp. 2990-2999. 2016.C
"Group equivariant convolutional networks". (2016)
Cite as: https://arxiv.org/abs/1602.07576

[3] Bekkers, E.J., Lafarge, M.W., Veta, M., Eppenhof, K.A., Pluim, J.P. and Duits, R., 2018, September. Roto-translation covariant convolutional networks for medical image analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 440-448). Springer, Cham.
Cite as: https://arxiv.org/abs/1804.03393