RAMAIAH
Institute of Technology

Investigation of Rotational Equivariance of Deep Neural Networks

Submitted as part of Mini-project
(EC64)

Bachelor of Engineering
in
Electronics and Communication

Names of the students:

Sinchana S R	USN: 1MS17EC106
Soma Rohith	USN: 1MS17EC109
Sushmith Thuluva	USN: 1MS17EC119
Chiranthan K	USN: 1MS17EC135

Under the Guidance of

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

Department of Electronics and Communication

RAMAIAH INSTITUTE OF TECHNOLOGY
(Autonomous Institute, Affiliated to VTU)
Accredited by National Board of
Accreditation & NAAC with
'A' Grade
MSR Nagar, MSRIT Post, Bangalore – 560054

2020

CERTIFICATE

This is to certify that the project work titled “Investigation of Rotational Equivariance of Deep Neural Networks” was carried out by Sinchana S R(USN: 1MS17EC106), Soma Rohith(USN: 1MS17EC109), Sushmith Thuluva(USN: 1MS17EC119) and Chiranthan K(USN: 1MS17EC135), bonafide students of Ramaiah Institute of Technology, Bangalore, in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Electronics and Communication awarded by Visvesvaraya Technological University, Belagavi, during the year 2020 – 2021. It is certified that all corrections/suggestions indicated during Internal Assessment have been incorporated in the report. The report has been approved as it satisfies the academic requirements prescribed.

(Signature of the Guide)
Dr. Raghuram S

(Signature of the HoD)
Dr. S Sethu Selvi

Name & Signature of Examiners with Date:-

1)

2)

DECLARATION

We hereby declare that the Project entitled “Investigation of Rotational Equivariance of Deep Neural Network” has been carried out independently at Ramaiah Institute of Technology under the guidance of Dr. Raghuram S, Associate Professor, Department of Electronics and Communication, RIT, Bangalore.

Signature of Students:

1. Sinchana S R USN: 1MS17EC106
2. Soma Rohith USN: 1MS17EC109
3. Sushmith Thuluva USN: 1MS17EC119
4. Chiranthan K USN: 1MS17EC135

Place:
Date:

Acknowledgment

The immense satisfaction that accompanies the successful completion of the project would be incomplete without the mention of the people who made it possible. We consider it our honor to express our deepest gratitude and respect to the following people who always guided and inspired us during the course of the Project.

We are deeply indebted to **Dr. N V R Naidu**, Principal, RIT, Bangalore for providing us with a rejuvenating program under a very creative learning environment.

We are much obliged to **Dr. S Sethu Selvi**, Professor &HOD, Department of Electronics and Communication, RIT, Bangalore for her constant support and motivation.

We sincerely thank our guide **Dr. Raghuram S**, Associate Professor, Department of Electronics and Communication, RIT, Bangalore, and express our humble gratitude for his valuable guidance, inspiration, encouragement, and immense help which made this project work a success.

We sincerely thank all the faculty members of the Department of E&C, RIT for their kind support to carry out this project successfully. Last but not least we would like to express our heartfelt gratitude to our parents, relatives, and friends for their constant support, motivation, and encouragement.

ABSTRACT

The main focus of this project is to explore the rotational equivalent capabilities of CNN's and making the feature extraction orientation independent, which is very essential in many cases.

Concepts like Group Convolutional layers, for the network to learn feature maps associated with different rotated versions of the input image in a single pass and Special Euclidean group (SE-2), for creating the desired lifting and group convolution layers that are SE(2) covariant and thus making the output roto-translate with the input, have been utilized for achieving the required result for the dataset.

The dataset selected for the project is the digital histopathological images for cancer detection.

With successful implementation and verification of the concepts used on the dataset, the efficiency of the proposed model can be compared with the conventional CNNs and thus proving the necessity of rotational equivariance for CNN's.

ACRONYMS

CNN	Convolutional Neural Networks
G-CNN	Grouped-Convolution Neural Network
SE(2)	Special Euclidean motion group
PCam	PatchCamelyon
ReLU	Rectified Linear Unit
G-Convolution	Group Convolution
AlexNet	Kind of neural network
GPU	Graphical Processing Unit
2-D	2-Dimension
WSI	Whole-side image
Acc	Accuracy
AUC	Area Under the Curve
AAM	Adam Adaptive Momentum
CIFAR	Canadian Institute For Advanced Research
ResNet	Residual Network
3-D	3-Dimension
AMIDA	Assessment of Mitosis Detection Algorithm
ROC	Receiver Operator Characteristic curve
EM	Electron microscopy
ISBI	International Symposium on Biomedical Imaging

Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Convolutional Neural Networks

Convolutional neural networks(CNN) have gained immense popularity over the past decade mainly because they have proven to be very powerful models of sensory data such as images, video, etc. One primary reason behind this is that convolutional layers used in a deep network are translation equivalent. By translational equivalence, It means, that if an input image is shifted in any direction and fed to the deep net, it is equivalent to feeding the original non-shifted image to the deep net and then shifting the resulting feature maps. In other words, feature extraction is independent of spatial position. However, this is not the case for other transformations such as rotation. In the following article one of the major drawbacks of CNN i.e, their incapability to being rotationally equivalent is discussed.

1.1.1 Convolution in Convolutional Neural Networks

The convolutional neural network, or CNN for short, is a specialized type of neural network model designed for working with two-dimensional image data, although they can be used with one-dimensional and three-dimensional data. Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a “convolution“.

In the context of a convolutional neural network, convolution is a linear operation that involves the multiplication of a set of weights with the input, much like a traditional neural network. Given that the technique was designed for two-dimensional input, the multiplication is performed between an array of input data and a two-dimensional array of weights, called a filter or a kernel.

The filter is smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product. A dot product is an element-wise multiplication between the filter-sized patch of the input and filter, which is then summed, always resulting in a single value. Because it results in a single value, the operation is often referred to as the “scalar product“.
Using a filter smaller than the input is intentional as it allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input. Specifically, the filter is applied systematically to each overlapping part or filter-sized patch of the input data, left to right, top to bottom.

This systematic application of the same filter across an image is a powerful idea. If the filter is designed to detect a specific type of feature in the input, then the application of that filter systematically across the entire input image allows the filter an opportunity to discover that feature anywhere in the image. This capability is commonly referred to as translation invariance, e.g. the general interest in whether the feature is present rather than where it was present.

The output from multiplying the filter with the input array one time is a single value. As the filter is applied multiple times to the input array, the result is a two-dimensional array of output values that represent the filtering of the input. As such, the two-dimensional output array from this operation is called a “feature map“.

Once a feature map is created, we can pass each value in the feature map through a nonlinearity, such as a ReLU, much like we do for the outputs of a fully connected layer.

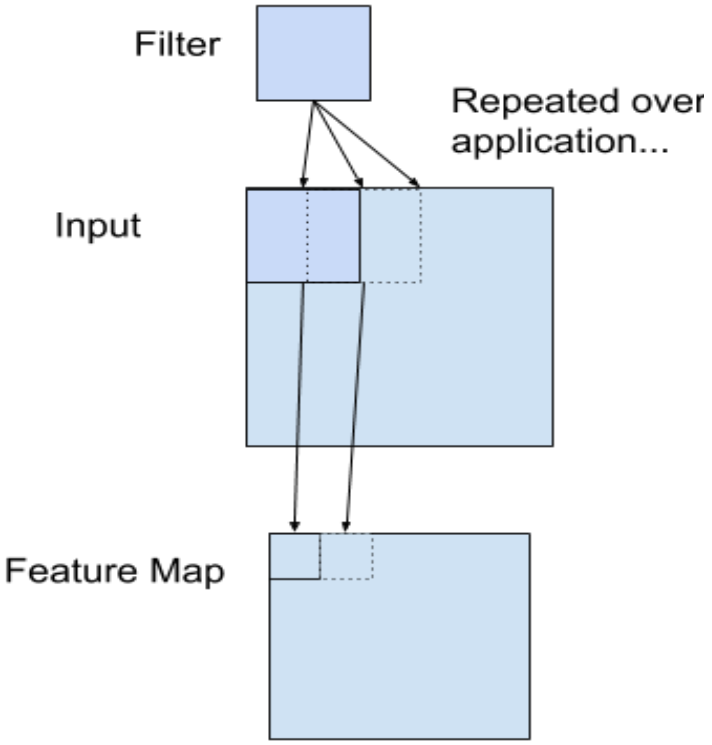


Fig 1.1 - Example of a Filter Applied to a Two-Dimensional Input to Create a Feature Map

If you come from a digital signal processing field or related area of mathematics, you may understand the convolution operation on a matrix as something different. Specifically, the filter (kernel) is flipped prior to being applied to the input. Technically, the convolution as described in the use of convolutional neural networks is actually a “cross-correlation”. Nevertheless, in deep learning, it is referred to as a “convolution” operation.

In summary, we have an input, such as an image of pixel values, and we have a filter, which is a set of weights, and the filter is systematically applied to the input data to create a feature map.

1.1.2 Convolution in Computer Vision

The idea of applying the convolutional operation to image data is not new or unique to convolutional neural networks; it is a common technique used in computer vision.

Historically, filters were designed by hand by computer vision experts, which were then applied to an image to result in a feature map or output from applying the filter then makes the analysis of the image easier in some way.

For example, below is a hand crafted 3×3 element filter for detecting vertical lines:

1) 0.0,1.0,0.0
2) 0.0,1.0,0.0
3) 0.0,1.0,0.0

Applying this filter to an image will result in a feature map that only contains vertical lines. It is a vertical line detector.
You can see this from the weight values in the filter; any pixel values in the center vertical line will be positively activated and any on either side will be negatively activated. Dragging this filter systematically across pixel values in an image can only highlight vertical line pixels.

A horizontal line detector could also be created and also applied to the
image, for example:

4) 0.0,1.0,0.0
5) 0.0,1.0,0.0
6) 0.0,1.0,0.0

Combining the results from both filters, e.g. combining both feature maps, will result in all of the lines in an image being highlighted.
A suite of tens or even hundreds of other small filters can be designed to detect other features in the image.

The innovation of using the convolution operation in a neural network is that the values of the filter are weights to be learned during the training of the network.

The network will learn what types of features to extract from the input. Specifically, training under stochastic gradient descent, the network is forced to learn to extract features from the image that minimize the loss for the specific task the network is being trained to

solve, e.g. extract features that are the most useful for classifying images as dogs or cats.

In this context, you can see that this is a powerful idea

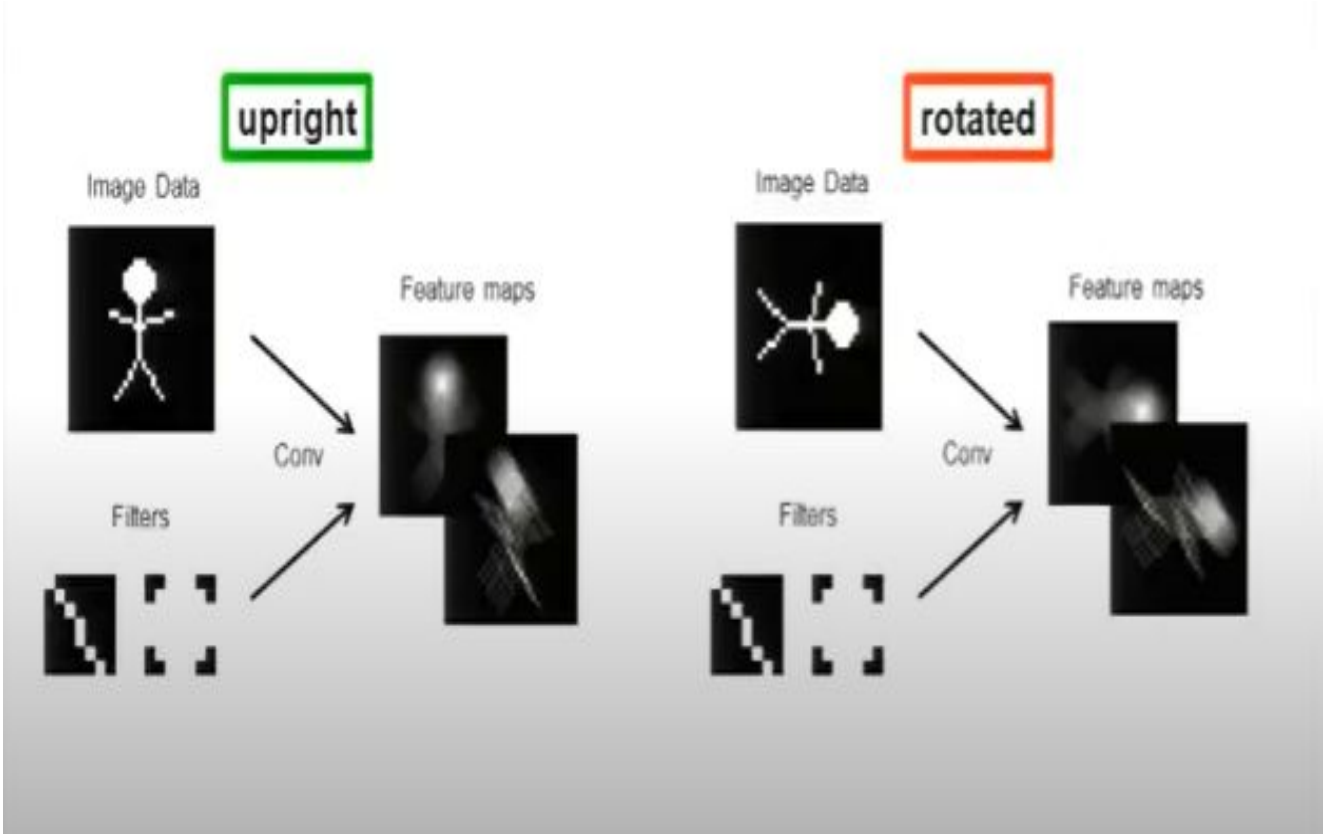


Figure 1.2 - Rotation of input image by 90 degrees

1.2 Overview

For example, if the input image is rotated by a certain degree and passed through a deep network then the resulting feature maps obtained are different from the feature maps obtained bypassing the original input image through a deep network and then rotating the feature maps. This implies that standard convolutional layers are not rotationally equivalent.

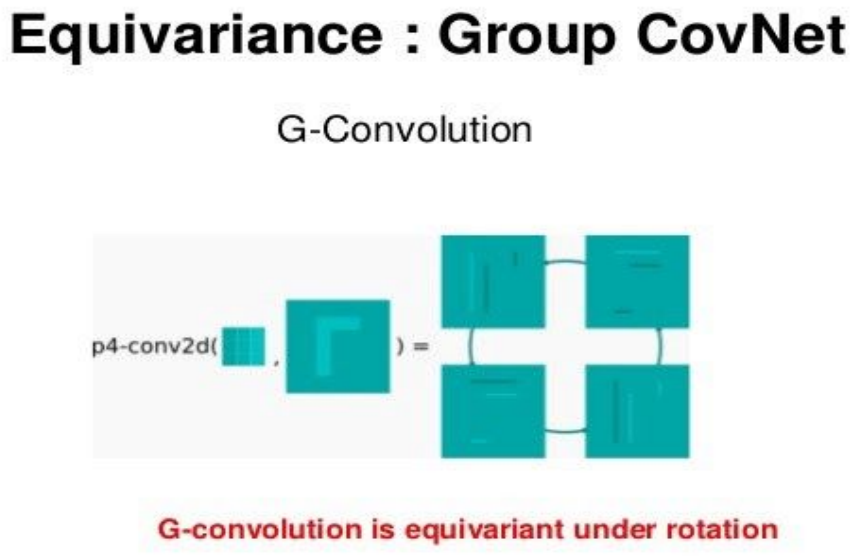


Figure 1.3 - Equivariant property

1.3 The need for rotational equivalence?

Today CNN's are the go-to solution for any task related to image processing, feature learning, etc. For many types of images, it is desirable to make feature extraction orientation independent as well. Typical examples are biomedical microscopy images or astronomical data which do not show a prevailing global orientation. Therefore it is a must that models designed for these data perform well even if there is a change in the orientation of the image given to the network. Thus, there must be some way in which this drawback of CNN could possibly be overcome.

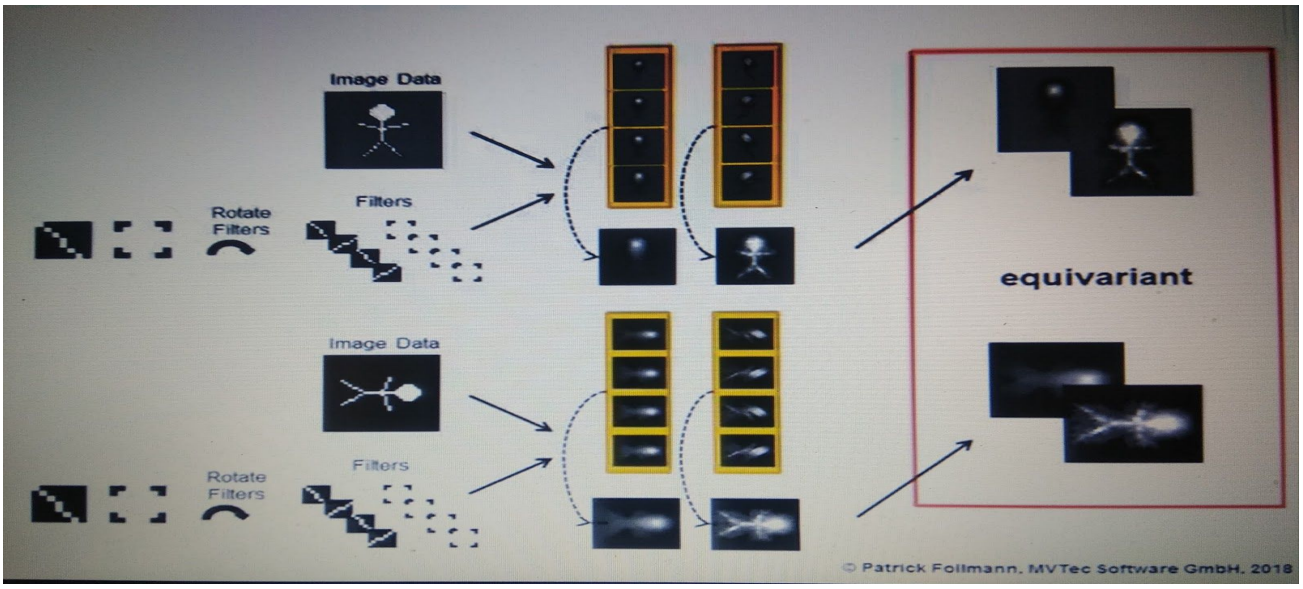


Figure 1.4 - Representation of G-CNN

1.4 Solution

Convolutional networks can be generalized to exploit larger groups of symmetries, including rotations and reflections. The possible way of achieving this is by using a new type of convolution layer called Group Convolution Layer instead of a standard planar Convolution layer.

A Group Convolution Layer can be understood by comparing it with planar Convolution layer, in planar Convolution layer we translate the filter and compute the inner product similarly a Group Convolution Layer can be viewed as a process in which we 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.

1.5 Grouped Convolution

Grouped convolution was introduced in the AlexNet paper in 2012. The main reason for implementing it was to allow the network training over two GPUs with limited memory (1.5 GB memory per GPU). The AlexNet below shows two separate convolution paths at most of the layers. It's doing model-parallelization across two GPUs (of course one can do multi-GPUs parallelization if more GPUs are available).

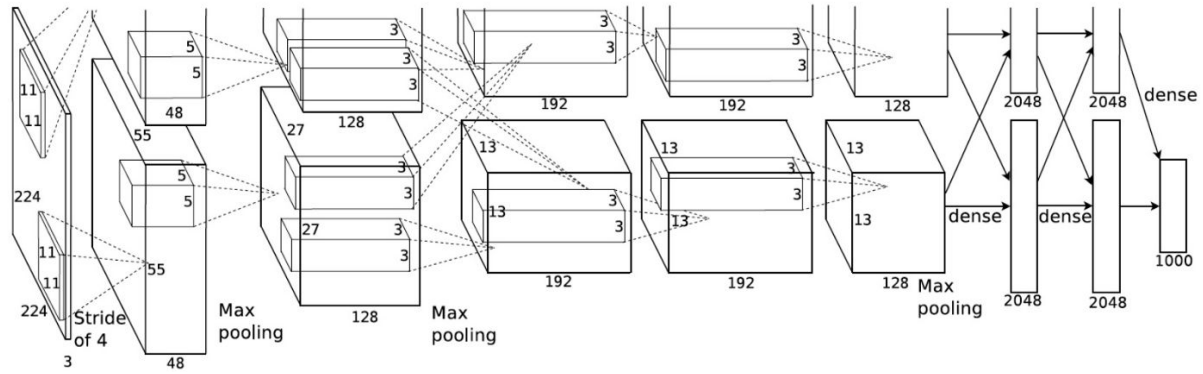


Figure 1.5 - Working of GCNN

Here we describe how the grouped convolutions work. First of all, conventional 2D convolutions follow the steps showing below. In this example, the input layer of size (7 x 7 x 3) is transformed into the output layer of size (5 x 5 x 128) by applying 128 filters (each filter is of size 3 x 3 x 3). Or in the general case, the input layer of size (Hin x Win x Din) is transformed into the output layer of size (Hout x Wout x Dout) by applying Dout kernels (each is of size h x w x Din)

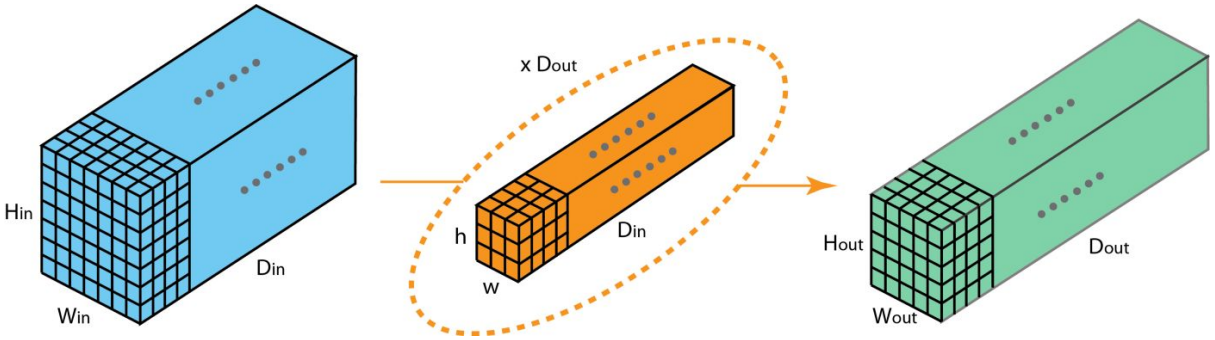


Figure 1.6 - Standard 2-D Convolution

In grouped convolution, the filters are separated into different groups. Each group is responsible for conventional 2D convolutions with a certain depth. The following examples can make this clearer.

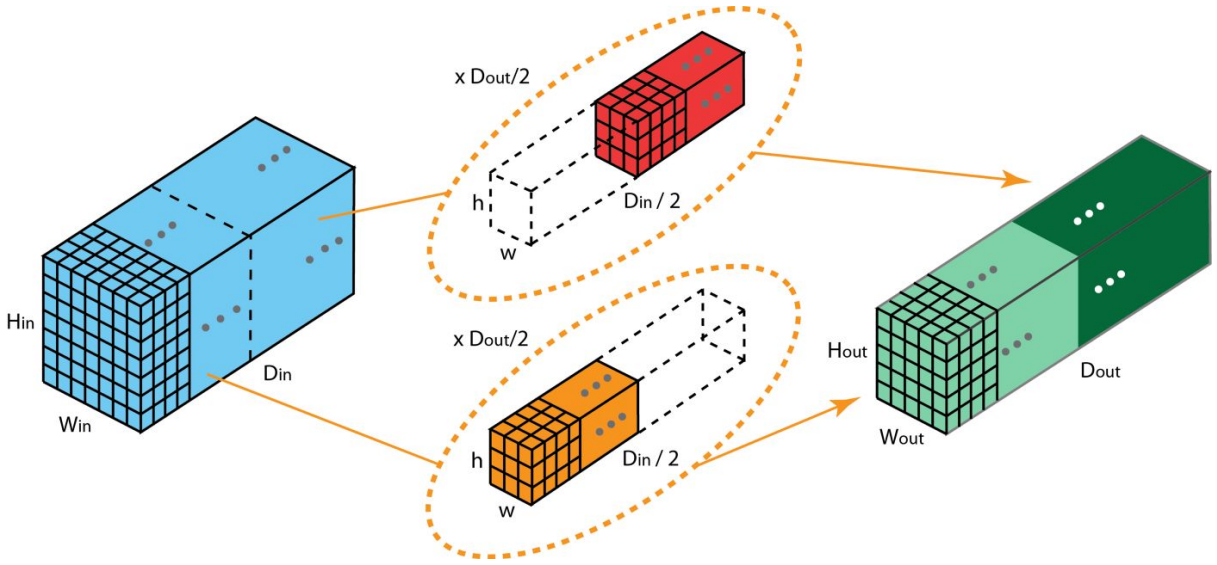


Figure 1.7 - Grouped Convolution with 2 Filter groups

Above is the illustration of grouped convolution with 2 filter groups. In each filter group, the depth of each filter is only half of that in the nominal 2D convolutions. They are of depth Din / 2. Each filter group contains Dout /2 filters. The first filter group (red) convolves with the first half of the input layer ([:, :, 0:Din/2]), while the second filter group (blue) convolves with the second half of the input layer ([:, :, Din/2:Din]). As a result, each filter group creates Dout/2 channels. Overall, two groups create 2 x Dout/2 = Dout channels. We then stack these channels in the output layer with Dout channels

1.6 Benefits of Group Convolution Layer

It can simply be used by replacing the standard convolution layer in a deep network.
Fast learning process
Allows the network to learn feature maps associated with different rotated versions of the input image in a single pass
Negligible computation overhead
Improves accuracy

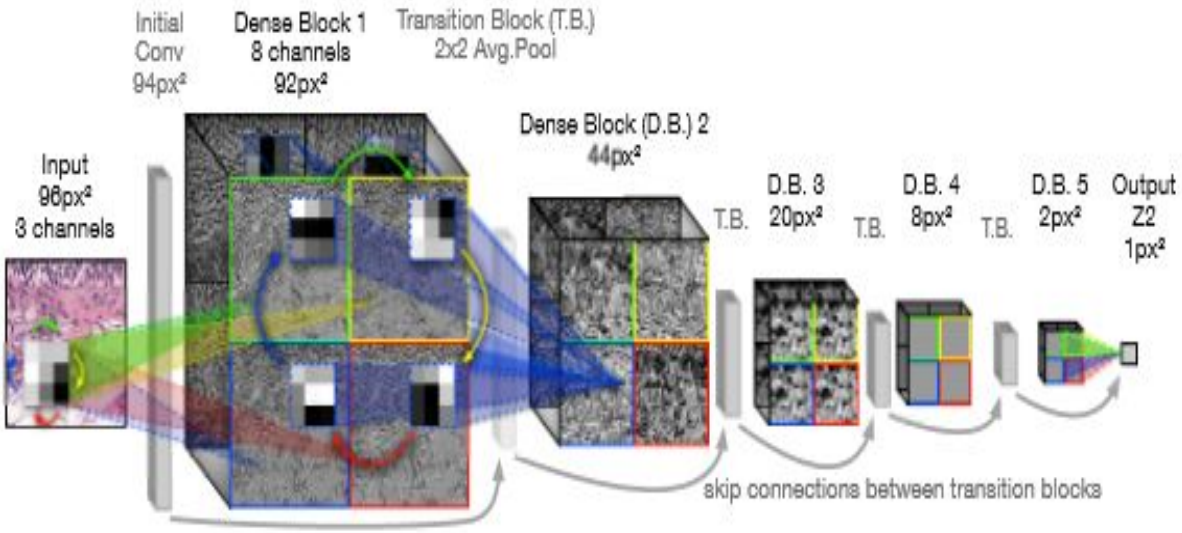


Figure 1.8 The proposed equivariant DenseNet architecture for the p4 group, consisting of 5 Dense Blocks (D.B.) alternated with Transition Blocks (T.B.). The final layer of the model is a p4 → Z2 group pooling layer followed by a sigmoid activation. The four orientations in p4 are illustrated through primary colors. A Z2 → p4 kernel (left),p4 → p4 kernel (middle), and p4 → Z2 kernel (right) illustrate how equivariance arises in the model.

Chapter 2: Problem statement

Deep Neural networks have been used in almost all fields for various applications because of its wide applicability, especially in the medical field. So,we are investigating the convolutional neural network model for detection of cancer cells in the histo-pathological image dataset(in general to all medical datasets) to overcome the drawback of rotational equivariance property in standard neural networks.This is being done by comparing the results of SE2CNN group convolution and standard CNN using a model like resnet50 or AlexNet, hence investigating the rotational equivariance property of deep neural networks.

Chapter 3: Literature review

3.1 Rotation equivariant CNNs for digital pathology[1]

3.1.1 Introduction

In this paper, a new model for digital pathology segmentation is proposed based on the observation that histopathology images are inherently symmetric under rotation and reflection. It presents the analysis that shows improved stability on predictions and demonstrates 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

3.1.2 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).

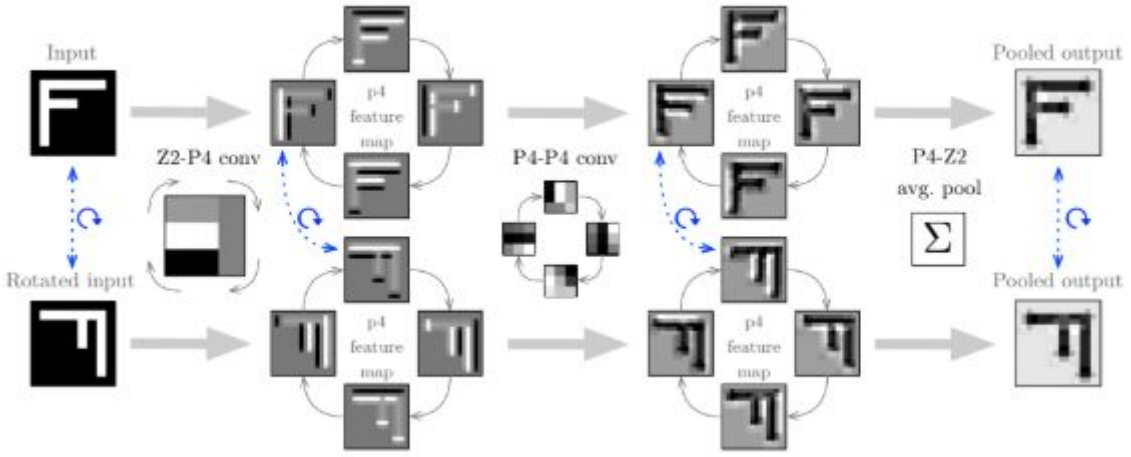


Figure 3.1 Given a canonical input and a rotated duplicate, we demonstrate how a 2-layer G-CNN is equivariant in p4. Feature maps of one kernel per layer are shown, and the dashed blue arrows indicate how (intermediate) representations of the two inputs correspond. The Z2 → p4 convolution correlates the input with 4 rotated versions of the same kernel. The p4 → p4 convolution correlates the resulting feature map with the p4-kernel, cyclically-shifting, and rotating the kernel for each orientation. The final layer demonstrates how average-pooling over the orientations produces a representation that is locally invariant and globally equivariant to the rotation. Global average pooling over p4 would result in a representation globally invariant to both translation and rotation.

3.1.3 Model architecture

The architecture is based on densely connected CN, which consists of dense blocks with layers that use the stack of all previous layers as input, alternated with transition blocks consisting of a 1x1 convolution layer and 2x2 stridden 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 a group-pooling layer followed by sigmoid activation.

3.1.4 Data Set Implementation

3.1.4.1 PatchCamelyon (PCam)

We assess the performance of our main contribution, the P4M-DenseNet architecture, on the PCam dataset. Table 1 reports the performance. P4M-DenseNet outperforms other models, closely followed by the P4-DenseNet, indicating that both rotation and reflection are useful inductive biases that can not be learned by data augmentation alone. Keeping the number of Z^2 maps fixed in the baseline degrades performance further, demonstrating the sample-efficiency of the P4M model.

3.1.4.2 Camelyon16

Table 1: Performance on PCam, measured by negative log-likelihood, accuracy and AUC. Experiments with additional data augmentation with 90° rotations and reflections are marked by +. M indicates matching number of $\mathbb{Z}^2$ maps, $\#W$ number of weights, K number of $\mathbb{Z}^2$ maps per layer.

Network	K	$\#W$	NLL	Acc	AUC
P4M-DenseNet	64	119K	**0.260**	**89.8**	**96.3**
P4M-DenseNet M	24	19K	0.273	89.3	95.8
P4-DenseNet	48	125K	0.329	89.0	94.5
DenseNet+	24	128K	0.306	88.1	95.1
DenseNet+ M	64	902K	0.365	87.2	94.6
DenseNet	24	128K	0.315	87.6	95.5

Figure 3.2 - Figure of Table 1

We evaluate our patch-based model on the slide-level tumor localization task of the Camelyon16 challenge. Fig. 4 reports the performance on the FROC score, next to those of a pathologist and the state-of-the-art approaches reported on this dataset. For the baseline DenseNet, the

training data is augmented with 90° rotations and reflection. We experiment with multiple data regimes, where the number of WSIs in the training set is incrementally reduced by a factor of two.

The results indicate that the proposed method performs consistently better than all compared methods in terms of the FROC metric. Comparing to the baseline DenseNet results, we see that the superiority of our proposed architecture is predominantly due to the increased parameter sharing by the p4m-equivariance, which frees up model capacity and reduces the redundancy of detecting the same histological patterns in different orientations.

We also observe that the performance gap between our model and the baseline increases when we limit the dataset size by removing WSIs. This seems to indicate that the performance in the small-data regime benefits significantly from the sample efficiency of P4M-DenseNet, with diminishing returns when the amount of data is sufficient for the baseline network to achieve (approximate) rotation equivariance. This performance gap remains for the full data set.

Model	Data	FROC
P4M-DenseNet 123k params	100%	84.0 (75.5, 91.5)
	50%	81.5 (72.2, 89.3)
	25%	72.6 (58.7, 84.6)
	12.5%	60.7 (46.0, 74.1)
DenseNet 126k params	100%	81.7 (72.1, 90.3)
	50%	80.0 (69.3, 89.1)
	25%	71.0 (57.7, 82.0)
	12.5%	55.4 (42.6, 68.5)
Liu *et al.* [1] 10×	100%	79.3 (74.2, 84.1)
Wang *et al.* [17]	100%	80.7*
Pathologist [9]	–	73.3

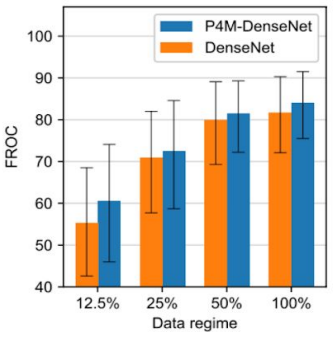


Figure 3.3 Performance on the Camelyon16 test set. The confidence bounds are obtained using a 2000-fold bootstrap regime. *Challenge winner [17] uses 40×resolution and is not directly comparable.

3.1.4.3 BreakHis

As an additional evaluation method, we assess the performance of the proposed model on the binary classification task of BreakHis. As training the model from scratch is impractical given the small dataset, we pre-train on Camelyon16 at a similar pixel resolution. We predict the

malignancy of a test image by using the maximum activation of 1000 random crops. We obtain an accuracy of 96.1 ± 3.2 and 93.5 ± 4.7 for

P4M-Densenet and the baseline respectively, outperforming previous approaches.

3.1.5 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 the reliability of the model. A derived patch-level dataset is presented, allowing straightforward and precise evaluation of a challenging histopathology task. It has been shown that the rotation equivariance improves the reliability of the model, motivating the application and further research of rotation equivariant models in the medical image analysis domain.

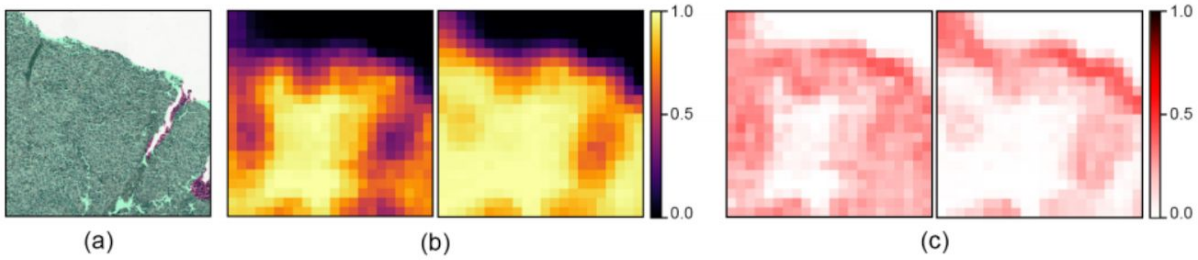


Figure 3.4 (a) shows a large input region spanning multiple patches, with the tumor ground truth overlayed in green. The region is predicted under 32 evenly spaced sub-90 rotations and prediction maps rotated back to the original orientation. (b) shows the mean prediction and (c) shows the standard deviation of the predictions across all rotations, using DenseNet (left) and P4M-DenseNet (right). Both networks are trained on the 12.5% data regime.

3.2 Group Equivariant Convolutional Networks[2]

3.2.1 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.

3.2.2 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.

3.2.3 Data Set Implementation

3.2.3.1 Rotated MNIST

The rotated MNIST dataset contains 62000 randomly rotated handwritten digits. The dataset is split into training, validation, and test sets of size 10000, 2000, and 50000, respectively.

We performed model selection using the validation set, yielding a

CNN architecture (Z2CNN) with 7 layers of 3 × 3 convolutions (4 × 4 in the final layer), 20 channels in each layer, relu activation functions, batch

normalization, dropout, and max-pooling after layer 2. For optimization, we used the Adam algorithm (Kingma & Ba, 2015). This baseline architecture outperforms the models tested by Larochelle et al. (2007) (when trained on 12k and evaluated on 50k) but does not match the previous state of the art, which uses prior knowledge about rotations (Schmidt & Roth, 2012).

Next, divided each convolution the number by a filter p4-convolution by $\sqrt{4} = 2$ (so as to keep the number of parameters approximately fixed), and added max-pooling over rotations after the last convolution layer. This architecture (P4CNN) was found to perform better without dropout, so we removed it. The P4CNN almost halves the error rate of the previous state of the art (2.28% vs 3.98% error).

We then tested the hypothesis that premature invariance is undesirable in a deep architecture. We took the Z2CNN, replaced each convolution layer by a p4- convolution followed by a coset max-pooling over rotations. The resulting feature maps consist of rotation- invariant features and have the same transformation law as the input image. This network (P4CNNRotationPooling) outperforms the baseline and the previous state of the art but performs significantly worse than
the P4CNN which does not pool over rotations in intermediate layers.

3.2.3.2 CIFAR-10

The CIFAR-10 dataset consists of 60k images of size 32 × 32, divided into 10 classes. The dataset is split into 40k training, 10k validation, and 10k testing split.

We compared the p4-, p4m- and standard planar Z^2 convolutions on two kinds of baseline architectures. Our first baseline is the All-CNN-C architecture by Springenberg et al. (2015), which consists of a sequence of 9 stridden and non-stridden convolution layers, interspersed with rectified linear activation units, and nothing else. Our second baseline is a residual network (He et al., 2016), which consists of an initial convolution layer, followed by three stages of2n convolution layers using k_i, followed by a final classification layer (6n + 2 layers in

total). The first convolution in each stage i > 1 uses a stride of 2, so the feature map sizes are 32, 16, and 8 for the three stages. For a constant

number of filters, this increases the size of the feature maps 4 or 8-fold, which in turn increases the number of parameters required per filter in the next layer. Hence, we halve it by roughly the number $\sqrt{8}$ of ≈ filters 3 in each in each p4m-Conv p4-Conv layer. layer, and This divide way, the number of parameters is left approximately invariant, while the size of the internal representation is increased. Specifically, we used k =11,23,45 for p4m-ResNet44. To evaluate the impact of data augmentation, we compare the networks on CIFAR10 and augmented CIFAR10+. The latter denotes moderate data augmentation with horizontal flips and small translations, following Goodfellow et al. (2013) and many others.

The training procedure for training the All-CNN has reproduced as closely as possible from Springenberg et al. (2015). For the ResNets, we used stochastic gradient descent with an initial learning rate of 0.05 and momentum 0.9. The learning rate was divided by 10 at epoch 50,100 and 150, and training was continued for 300 epochs.

3.2.4 Results and Conclusion

(As mentioned in the research paper)

By exploiting symmetries, G-CNN's achieve the 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.

3.2.5 Future Scope

In future work, we want to implement G-CNNs that work on hexagonal lattices which have an increased number of symmetries relative to square grids, as well as G-CNNs for 3D space groups. All of the theory presented in this paper is directly applicable to these groups, and the G-convolution can be implemented in such a way that new groups can

be added by simply specifying the group operation and a bijective map between the group and the set of indices.

One limitation of the method as presented here is that it only works for discrete groups. Convolution on continuous (locally compact) groups is mathematically well- defined, but may be hard to approximate in an equivariant manner. A further challenge, already identified by Gens & Domingos (2014), is that a full enumeration of transformations in a group may not be feasible if the group is large.

Finally, we hope that the current work can serve as a concrete example of the general philosophy of “structured representations”, outlined in section 2. We believe that adding mathematical structure to a representation (making sure that maps between representations preserve this structure), could enhance the ability of neural nets to see abstract similarities between superficially different concepts.

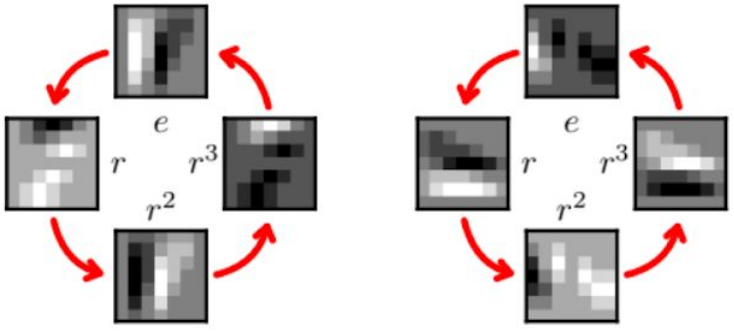


Figure 3.5: A p4 feature map and its rotation by r

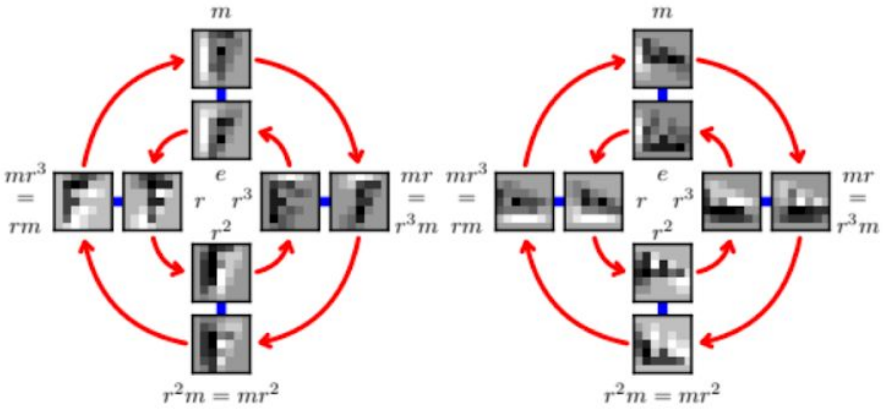


Figure 3.6: A p4m feature map and its rotation by r

3.3 Roto-Translation Covariant Convolutional Networks for Medical Image Analysis[3]

3.3.1 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.

3.3.2 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.

3.3.3 Data Set Implementation

3.3.3.1 Histopathology - Mitosis detection

The task aims at detecting mitotic figures in hematoxylin-eosin stained slides. We used the public dataset AMIDA13 that consists of high power field images from 23 breast cancer cases. Eight cases (458 mitoses) were used to train the networks with random batches of 68×68 image patches, balanced between mitotic and hard negative figures. This receptive field was obtained by means of max-pooling operations in the first three layers. Sets of candidate detections were generated as in After the selection of an operating point on four validation cases (92 mitoses). We assessed an $F_{1\text{-score}}$ for each model based on the 11 test cases (533 mitoses).

3.3.3.2 Retina - Vessel segmentation

In this task the blood vessels in the retina are segmented. For validation, we use the public DRIVE database, which consists of 40 retinal images with manual segmentations. The set is split between a training set (of which we use 16 for training, and 4 for validation) and a test set of also 20 images. The G-CNN's produce a probability for the vessel and background class. Training is done with 10000 patches (17 × 17) per class per image. The output probabilities can be thresholded to create a binary segmentation, which can be used to quantify performance in terms of sensitivity and specificity. The area under the receiver operator characteristic (ROC) curve, in short AUC, summarizes these performances into a single value.

3.3.3.3 Electron microscopy - Cell boundary segmentation

This task consists of segmenting the boundaries of cells that are imaged with EM. We use the data and evaluation system of the ISBI EM segmentation challenge. The data consists of 2 volumes (1 train, 1 test), each containing 30 consecutive images from a serial section transmission EM. Both the segmentation and the evaluation is done by treating the volumes as sequences of 2D slices. To increase receptive field size we include max-pooling in the first 2 layers. Training is done with 10000 patches (48 × 48) per class per image. The main evaluation criterion for the challenge is the Rand score, which measures the similarity between clusterings/connected components. The reported Rand score is the maximum score (for several thresholds) computed for the connected components obtained after thinning of the binary cell boundary segmentation.

3.3.4 Results and Conclusion (As mentioned in the research paper)

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≥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.

Images in Research Paper

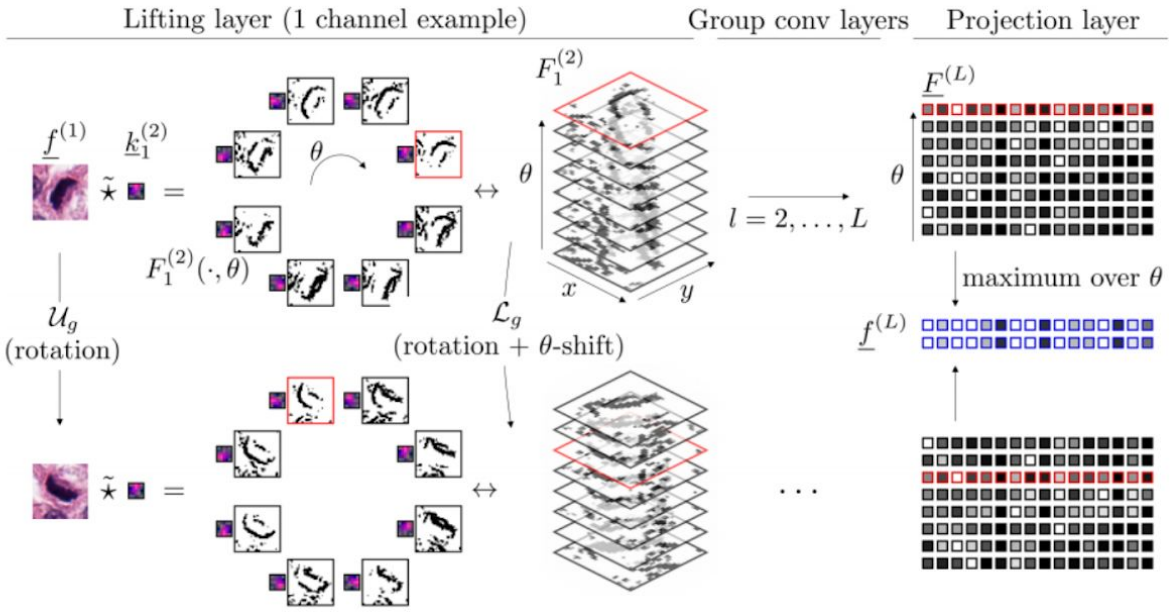


Figure 3.7 Rotation co- and invariance. Top row: the activations after the lifting convolutions with a single kernel k(2)1, stacked together it yields an SE(2) image F(2)1. The projection layer at the end of the pipeline gives a rotation-invariant feature vector.
Bottom row: the same figures with a rotated input.

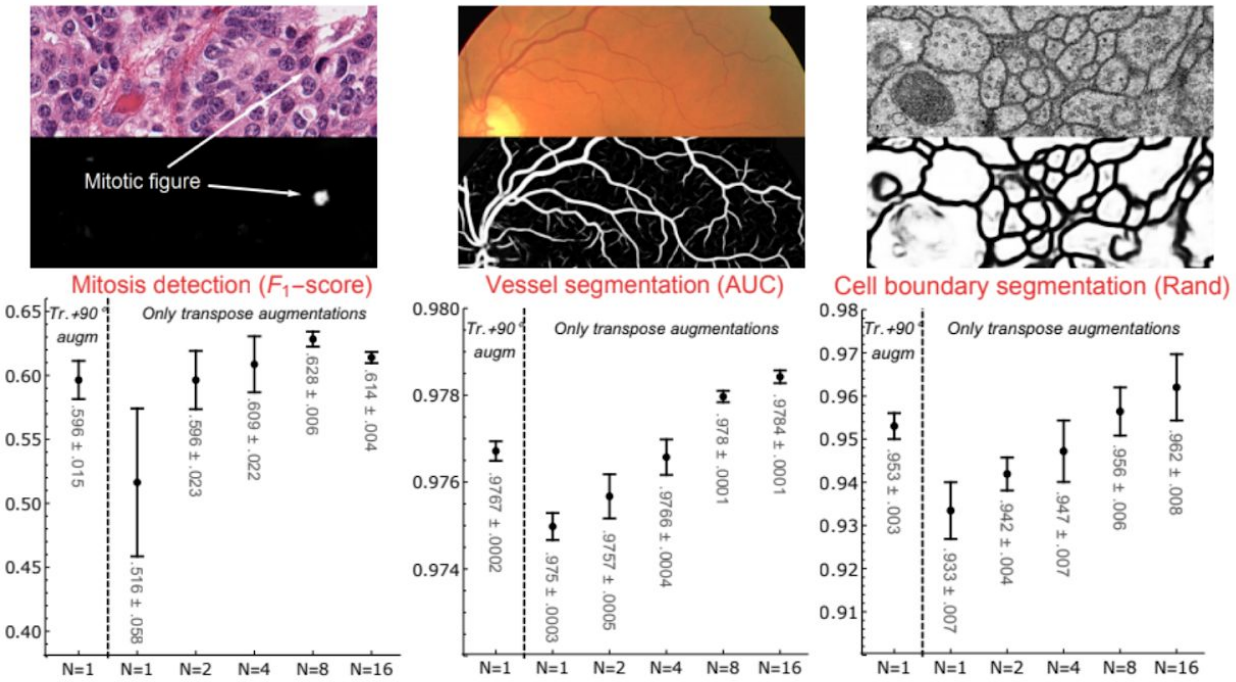


Figure 3.8 Top row: Crop outs of images of the three tasks with the class probabilities
generated by our method. Bottom row: Mean results (±1 std. dev.).

Chapter 4: 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 the results of SE2CNN group convolution and a standard CNN using a model like resnet50 or AlexNet. Comparing results of both to predict cancer, in turn investigating rotational equivariance of deep neural networks.

In SE(2) CNN, 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.

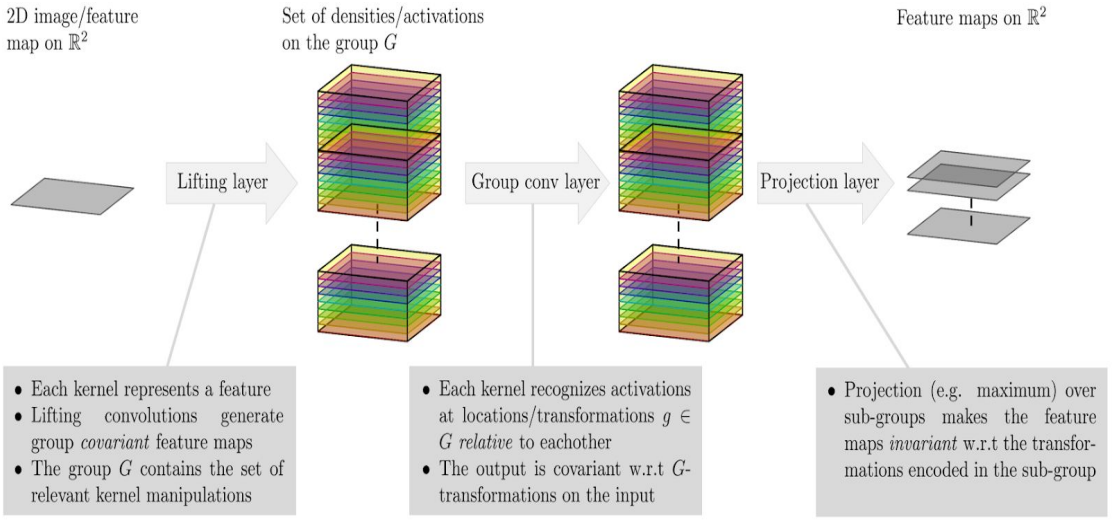


Figure 4.1 Group convolutional networks

In this case the lifting layer probes the 2D input with rotated and translated versions of the convolution kernels. The data is thus lifted to the space of positions and orientations. In order to make the following layers equivariant with respect to rotations and translations of the input, these layers are defined in terms of the left-regular representation of SE(2) on SE(2)-images. The kernels used in the layers are trained to recognize the (joint) activations of positions and orientations relative to each other. Finally, in order to make the entire network invariant to certain transformations, one can decide to apply max-pooling over sub-groups. In our case, we might for example do a maximum projection over the sub-group of rotations in order to make the network locally rotation invariant.

4.1 Libraries used

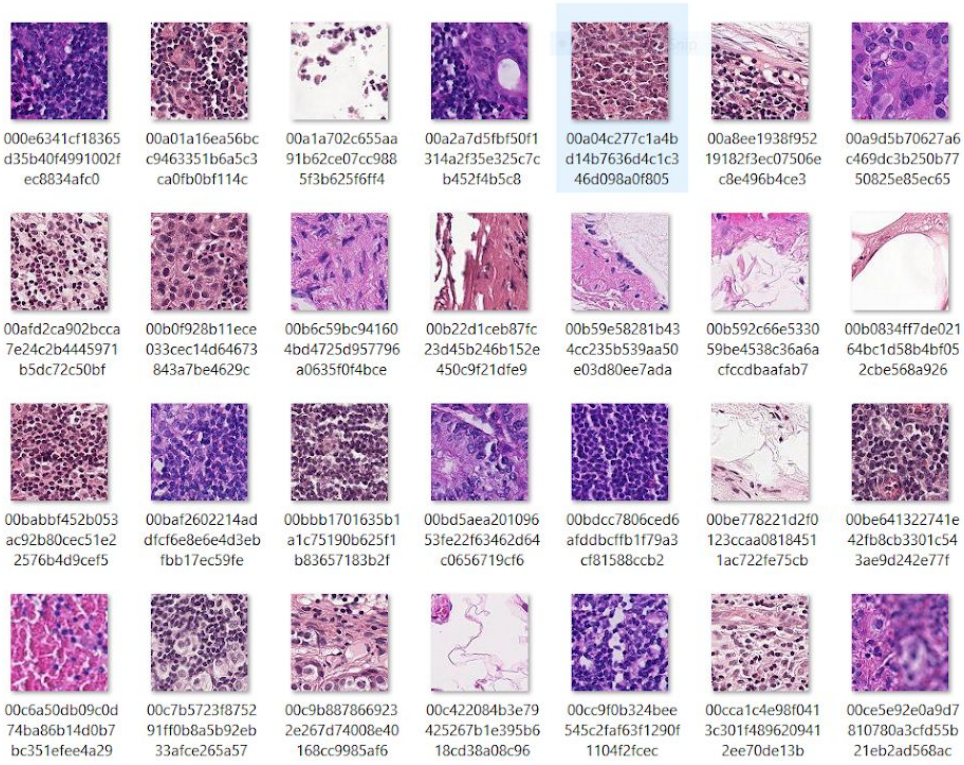
1. Keras https://github.com/keras-team/keras
2. Tensorflow https://github.com/tensorflow/tensorflow
3. SE2CNN https://github.com/tueimage/SE2CNN
   (For G-CNN [3])
4. MonkAI https://github.com/Tessellate-Imaging/monk_v1
   (For Standard CNN)

Chapter 5: Progress

We have processed the image and modified the dataset.

The original DATASET is https://www.kaggle.com/c/histopathologic-cancer-detection/data

Figure 5.1



The modified dataset contains images of the original dataset which is renamed, cropped to the 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 32*32px region of a patch contains at least one pixel of tumor tissue.

20% of train images of the original dataset are in the validation folder 80% of train images of the original dataset

are in the train folder All images of the test folder of the original dataset are in the test folder.

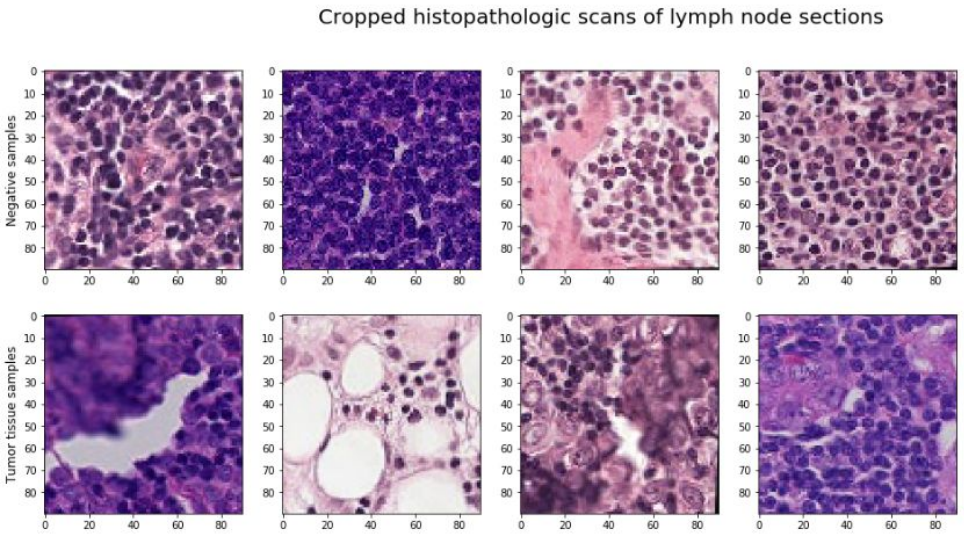


Figure 5.2

Both the train and validation folder of this modified dataset has CANCER and Normal folder images 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 the 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% for G-CNN and around 79% for standard CNN using MONK

Chapter 6: Results and Discussion

6.1 Image Processing/ Data Preprocessing

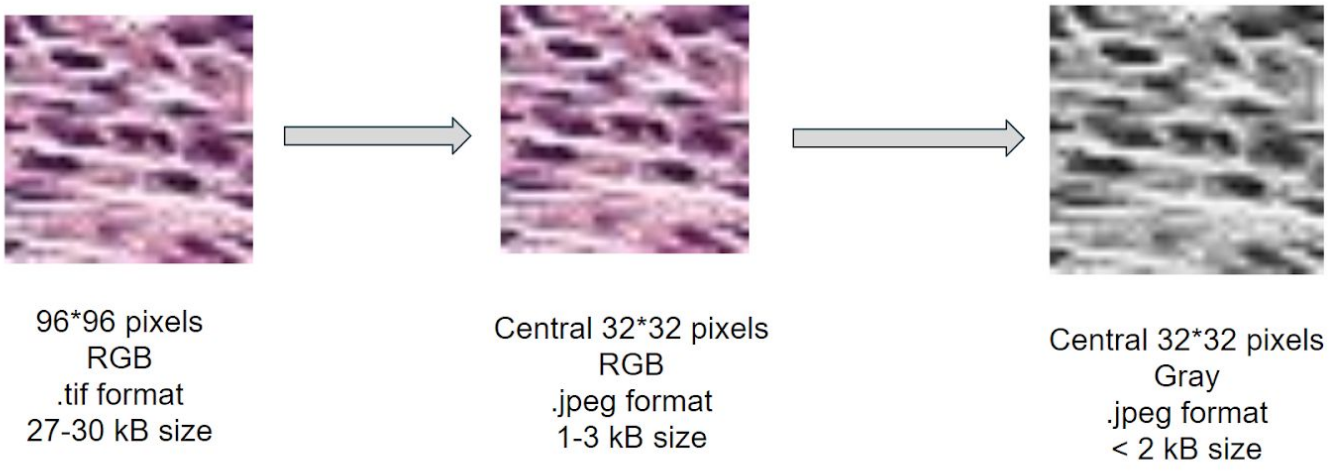


Figure 6.1: Image Processing

6.2 Results of SE2CNN G-CNN on Cancer Dataset

SE2CNN model (group convolution) having only 5 layers over 10 epochs gave an accuracy of 59.825% with an error rate of 40.17498

https://www.kaggle.com/sinchubhat/histopathalogic-cancer-detection

```
Epoch  0  finished... Average loss =  0.8274 , time =  1191.6
782
Epoch  1  finished... Average loss =  0.6755 , time =  1200.7
101
Epoch  2  finished... Average loss =  0.6754 , time =  1162.3
236
Epoch  3  finished... Average loss =  0.6755 , time =  1157.1
764
Epoch  4  finished... Average loss =  0.6754 , time =  1147.1
641
Epoch  5  finished... Average loss =  0.6754 , time =  1151.3
223
Epoch  6  finished... Average loss =  0.6755 , time =  1157.3
637
Epoch  7  finished... Average loss =  0.6755 , time =  1143.4
Epoch  8  finished... Average loss =  0.6754 , time =  1143.6
563
Epoch  9  finished... Average loss =  0.6754 , time =  1144.9
942
```

```
print(' Compare the first 10 results with the ground truth ')
print(' The first 10 predicted results ')
print(labels_pred[0:10])
print(' The first 10 ground truth (actual results) ')
print(eval_labels[0:10])
```

```
 Compare the first 10 results with the ground truth
 The first 10 predicted results
[0 0 0 0 0 0 0 0 0 0]
 The first 10 ground truth (actual results)
[1. 0. 0. 0. 1. 0. 0. 0. 0. 1.]
```

```
print(' The accuracy (average number of successes) ')
print(((labels_pred - eval_labels)**2==0).astype(float).mean())
```

```
 The accuracy (average number of successes)
0.5982501988410408
```

```
print(' Total number of errors ')
print(((labels_pred - eval_labels)**2>0).astype(float).sum())
```

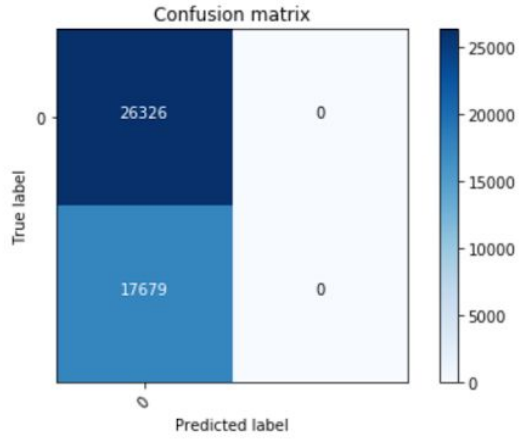
```
 Total number of errors
17679.0
```

```
print(' Error rate ')
print(100*((labels_pred - eval_labels)**2>0).astype(float).mean
())
```

```
 Error rate
40.17498011589592
```

```
cm = confusion_matrix(eval_labels, labels_pred)
plot_confusion_matrix(cm, range(1))
```

```
Confusion matrix, without normalization
```



6.3 Results of Standard CNN Using MonkAI on Cancer Dataset

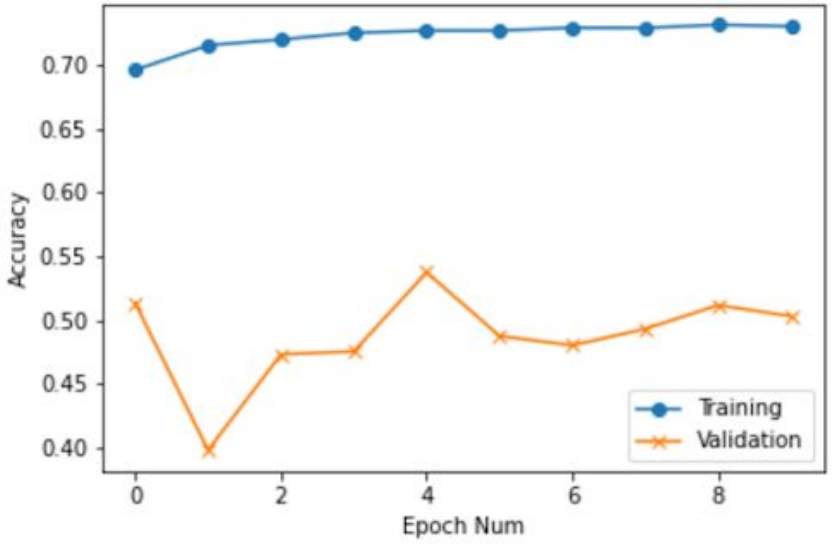
The results of standard CNN using MonkAI is as follows

Model - resnet152_v2 ,Keras backend, 10 epochs, average accuracy of 50.79422792864447% (version-6 of the notebook)

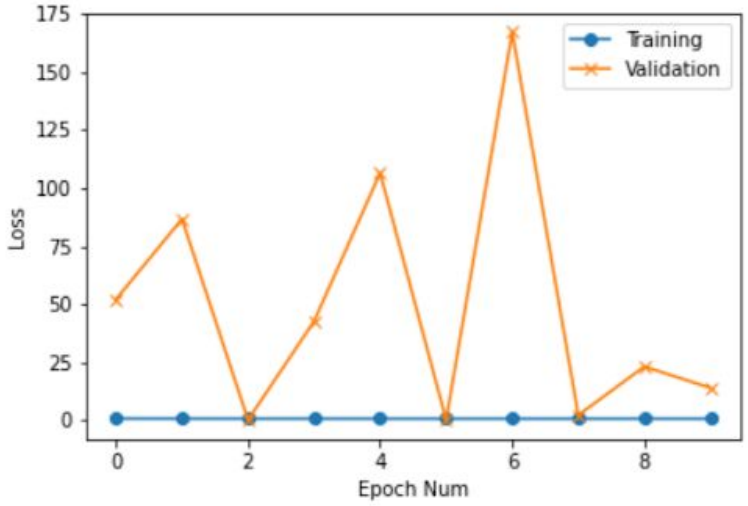
model - RESNET50, Keras backend, 10 epochs, having 108 layers in model with 2 trainable layers gave an average accuracy of 60.1795% (version-3 of the notebook)

https://www.kaggle.com/sinchubhat/cancer-detection-using-monk

Train Validation Accuracy Curve



Train Validation Loss Curve



6.4 Results of Standard CNN Using Keras on Cancer Dataset

A simple CNN using Keras having 5 layers over 65 epochs gave a training accuracy of 79.35% and validation accuracy of 77.09%

https://www.kaggle.com/sinchubhat/cnn-cancer

```
Epoch 22/25
8000/8000 [==============================] - 521s 65ms/step -
loss: 0.4618 - accuracy: 0.7898 - val_loss: 0.4388 - val_accur
acy: 0.7609
Epoch 23/25
8000/8000 [==============================] - 519s 65ms/step -
loss: 0.4614 - accuracy: 0.7900 - val_loss: 0.5342 - val_accur
acy: 0.7526
Epoch 24/25
8000/8000 [==============================] - 511s 64ms/step -
loss: 0.4619 - accuracy: 0.7901 - val_loss: 0.5680 - val_accur
acy: 0.7624
Epoch 25/25
8000/8000 [==============================] - 515s 64ms/step -
loss: 0.4612 - accuracy: 0.7901 - val_loss: 0.5346 - val_accur
acy: 0.7732
```

```
Epoch 27/30
2000/2000 [==============================] - 147s 73ms/step -
loss: 0.4601 - accuracy: 0.7923 - val_loss: 0.5055 - val_accur
acy: 0.7698
Epoch 28/30
2000/2000 [==============================] - 148s 74ms/step -
loss: 0.4575 - accuracy: 0.7904 - val_loss: 0.4367 - val_accur
acy: 0.7694
Epoch 29/30
2000/2000 [==============================] - 145s 73ms/step -
loss: 0.4588 - accuracy: 0.7915 - val_loss: 0.5711 - val_accur
acy: 0.7611
Epoch 30/30
2000/2000 [==============================] - 148s 74ms/step -
loss: 0.4545 - accuracy: 0.7935 - val_loss: 0.5175 - val_accur
acy: 0.7709
```

6.5 Results of Standard CNN Using MonkAI on Pneumonia Dataset

Model - resnet50, Pytorch backend, 25 epochs, the average accuracy of 100%

https://www.kaggle.com/sinchubhat/pneumonia-classification-monk

Run Validation

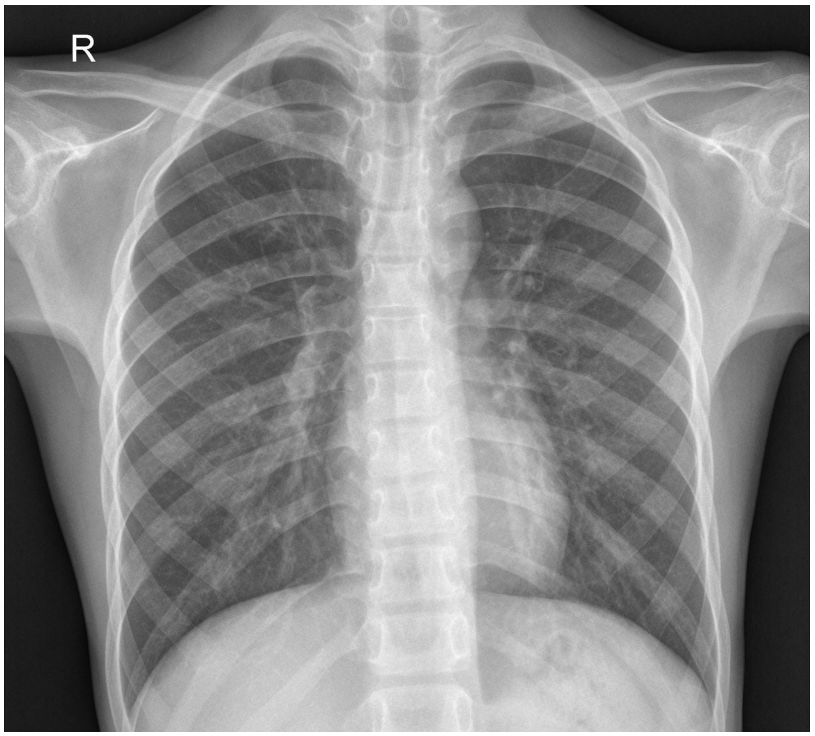
```
Result
    class based accuracies
        0. NORMAL - 100.0 %
        1. PNEUMONIA - 100.0 %
    total images:           16
    num correct predictions: 16
    Average accuracy (%):    100.0
```

Running inference on test Images

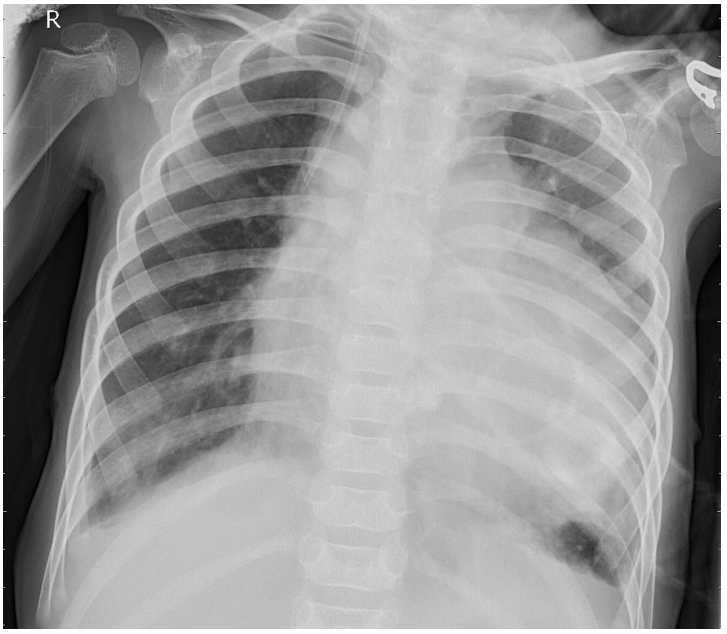
```
Pytorch Version: 1.4.0

Model Details
    Loading model - workspace/PneumoniaClassificationMONK/Usin
gPytorchBackend/output/models/final
    Model loaded!

Experiment Details
    Project: PneumoniaClassificationMONK
    Experiment: UsingPytorchBackend
    Dir: /kaggle/working/workspace/PneumoniaClassificationMON
K/UsingPytorchBackend/
```

Select Image and Run Inference

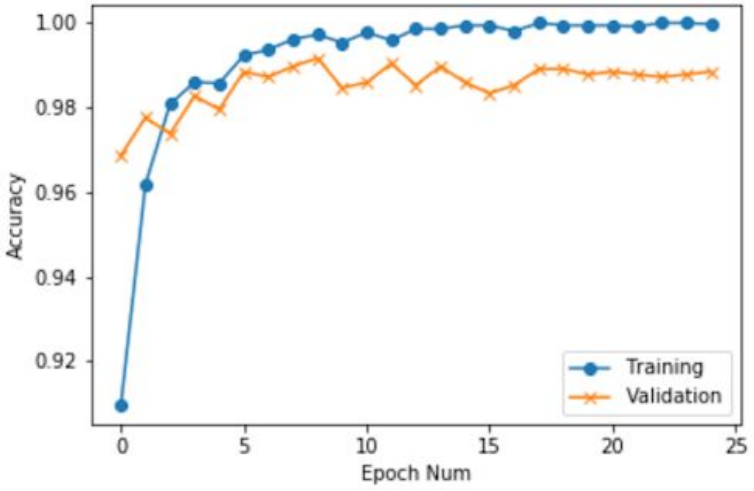
```
Prediction
    Image name:         /kaggle/input/chest-xray-pneumonia/che
st_xray/test/NORMAL/IM-0005-0001.jpeg
    Predicted class:      NORMAL
    Predicted score:      1.52426278591156
```



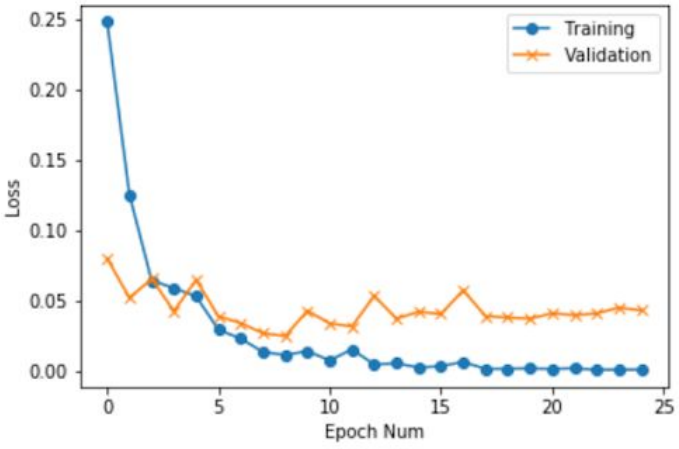

```
Prediction
    Image name:         /kaggle/input/chest-xray-pneumonia/che
st_xray/test/PNEUMONIA/person108_bacteria_506.jpeg
    Predicted class:      PNEUMONIA
    Predicted score:      7.050433158874512
```



Train Validation Accuracy curve



Train Validation Loss Curve



6.6 Results of G-CNN on Pneumonia Dataset

https://www.kaggle.com/sinchubhat/pneumonia-classification-se2cnn

Data Train Label

```
0       0
1       1
2       0
3       0
4       1
       ..
5211    0
5212    1
5213    1
5214    1
5215    0
Name: label, Length: 5216, dtype: int64
```

Data Validate Label

```
0     1
1     1
2     0
3     0
4     1
5     1
6     1
7     0
8     1
9     0
10    0
11    0
12    0
13    1
14    1
15    0
Name: label, dtype: int64
```

```
 Length of train_data
5216
 Length of eval_data
16
 Length of train_labels
5216
 Length of eval_labels
16
```

6.7 Results of Standard CNN Using MonkAI on Covid-19 Dataset

Model - densenet201, Keras backend, 25 epochs, best validation accuracy of 61.9718%

https://www.kaggle.com/sinchubhat/covid19-analysis-using-monk

```
Dataset Numbers
    Num train images: 176
    Num val images:   75
    Num classes:      3

Model Params
    Model name:             densenet201
    Use Gpu:                True
    Gpu Memory Fraction:    0.6
    Use pretrained:         True
    Freeze base network:    True
```

End of 25th epoch and validation accuracy

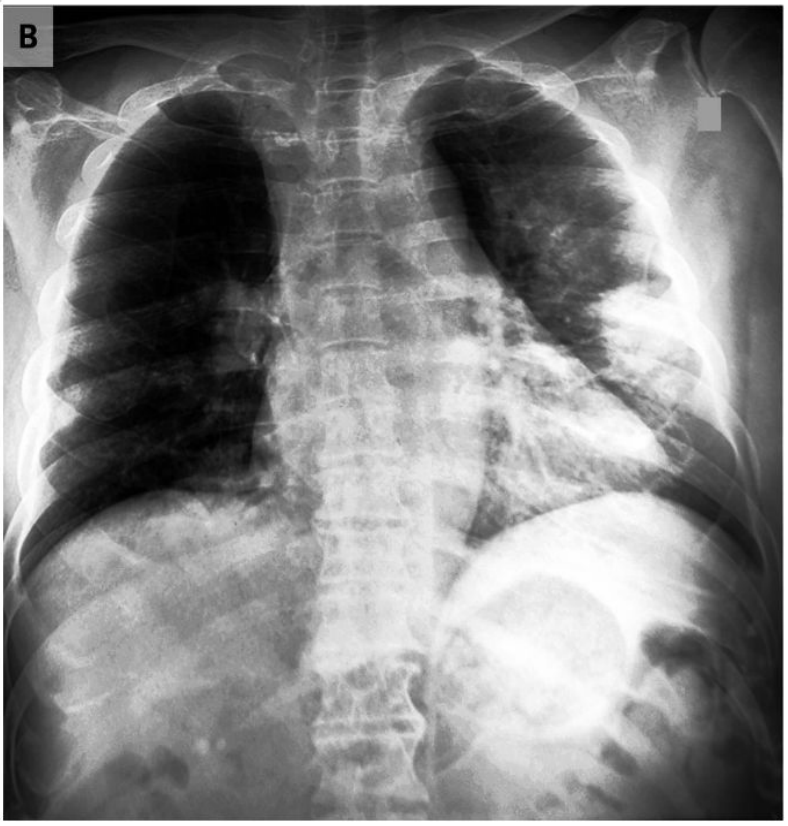
```
Epoch 00025: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-07.
    Training completed in: 9m 20s
    Best val Acc:          0.619718

Training End
```

results along with average accuracy

```
Result
    class based accuracies
        0. Covid - 15.384615384615385 %
        1. Normal - 100.0 %
        2. Viral Pneumonia - 25.0 %
    total images:           66
    num correct predictions: 29
    Average accuracy (%):    43.93939393939394
```

Prediction with single image input

```
Prediction
    Image name:         /kaggle/input/covid19-image-dataset/Covid19-dataset/test/Covid/0108.
jpeg
    Predicted class:      Covid
    Predicted score:      0.6674274206161499
```



```
Prediction
    Image name:         /kaggle/input/covid19-image-dataset/Covid19-dataset/test/Normal/012
1.jpeg
    Predicted class:      Normal
    Predicted score:      0.8929290771484375
```

6.8 Results of Standard CNN Using MonkAI on Lung and Colon Cancer Dataset

Lung Cancer Dataset

Model - resnet18, Pytorch backend, 5 epochs, best validation accuracy of 94.6889%

Colon Cancer Dataset

Model - resnet152, Pytorch backend, 5 epochs, the average accuracy of 99.1%

https://www.kaggle.com/sinchubhat/lung-colon-cancer-histopathological-images-monk

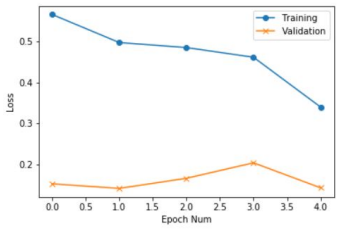
Number of dataset and model specification

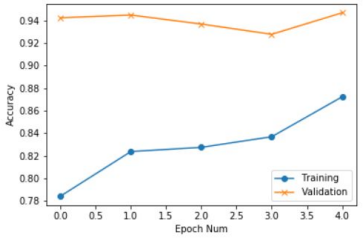
```
Dataset Numbers
    Num train images: 10500
    Num val images:   4500
    Num classes:      3

Model Params
    Model name:           resnet18
    Use Gpu:              False
    Use pretrained:       True
    Freeze base network:  True
```

Validation accuracy at the end of 5th epoch

```
    curr_lr - 0.001
    [Epoch 5] Train-acc: 0.872, Train-loss: 0.340 | Val-acc: 0.946889, Val-loss: 0.144,
| time: 956.0 sec

    Training completed in: 82m 1s
    Best val Acc:          0.946889
```

Accuracy and loss curves





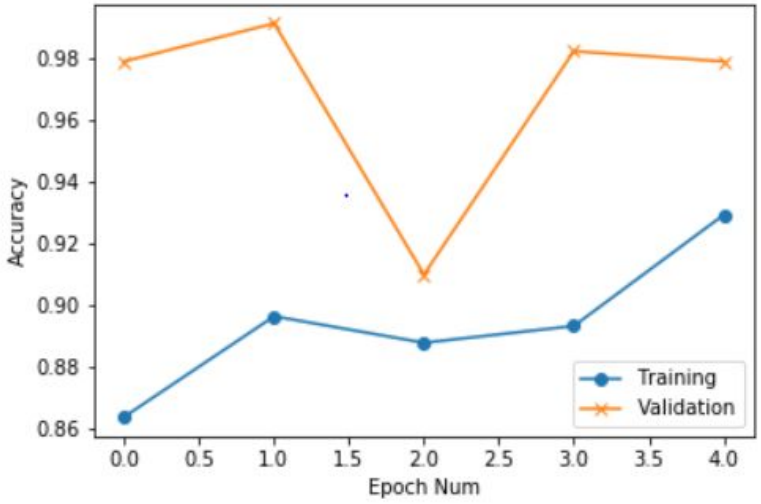
using resnet152:

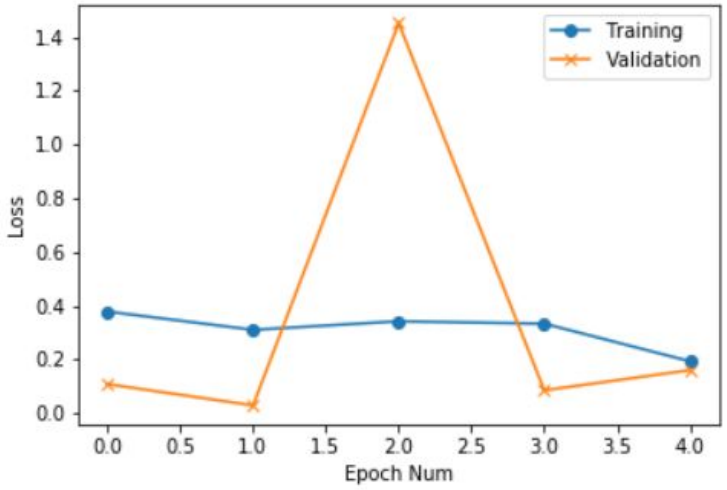
```
Dataset Numbers
    Num train images: 7000
    Num val images:   3000
    Num classes:      2

Model Params
    Model name:           resnet152
    Use Gpu:              False
    Use pretrained:       True
    Freeze base network:  True
```

Validation accuracy at the end of 5th epoch

```
    curr_lr - 0.001
    [Epoch 5] Train-acc: 0.929, Train-loss: 0.194 | Val-acc: 0.978667, Val-loss: 0.161,
| time: 3460.0 sec

    Training completed in: 300m 30s
    Best val Acc:          0.991000

Training End
```

Accuracy and loss curves:





Chapter 7: Conclusion and Future Scope

The model we developed can be deployed into smaller machinery in rural healthcare services as an alternative for high tech facilities.
The Rural areas having no sufficient infrastructure to support advanced and space-occupying facilities find themselves at a disadvantage in analyzing the medical conditions of their community.
In Remote areas, the low availability of manpower and insufficient background support makes it very difficult for the people living in the areas to assess their medical conditions.
In cases as above, our model can be deployed into optimized hardware to solve the drawbacks discussed.

The Developed model can also be applied to various orientation sensitive data sets in various scientific fields.
With the ever-increasing demand for automation, various scientific fields look up to machine learning for saving time and making the machine smarter as technology advances.
Fields like Space travel, Astronomy, Remote Sensing, Surveillance, and many other image processing sectors depend on Machine learning algorithms for efficient and faster analysis. The model we developed can be used in astronomy for detecting constellations and mapping the cosmic map in a better way.

As discussed in the report, the importance of deep neural networks is rapidly increasing in the field of medical analysis.
With ever-increasing and mutating diseases like the recent COVID-19 pandemic, it is becoming very essential for medical researchers to analyze various aspects of the disease and deep neural networks can play a vital role in aiding these investigations.

References

[1] Veeling, B. S., Linmans, J., Winkens, J., Cohen, T., & Welling, M. (2018, September). Rotation equivariant CNNs for digital pathology. *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. *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 440-448). Springer, Cham.
Cite as: https://arxiv.org/abs/1804.03393

[4] Dieleman, S., Willett, K. W., and Dambre, J. Rotation- invariant convolutional neural networks for galaxy mor- phology prediction. Monthly Notices of the Royal Astro- nomical Society, 450(2), 2015.

[5] Lenc, K. and Vedaldi, A. Understanding image represen- tations by measuring their equivariance and equivalence. In Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2015.

[6] Oyallon, E. and Mallat, S. Deep Roto-Translation Scat- tering for Object Classification. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2865—-2873, 2015.

[7] Schmidt, U. and Roth, S. Learning rotation-aware fea- tures: From invariant priors to equivariant descriptors. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[8] Sifre, Laurent and Mallat, Stephane. Rotation, Scaling and Deformation Invariant Scattering for Texture Discrimi- nation. IEEE conference on Computer Vision and Pat- tern Recognition (CVPR), 2013.

[9] Skibbe, H. Spherical Tensor Algebra for Biomedical Im- age Analysis. PhD thesis, Albert-Ludwigs-Universitat Freiburg im Breisgau, 2013.

[10] Zhang, C., Voinea, S., Evangelopoulos, G., Rosasco, L., and Poggio, T. Discriminative template learning in group-convolutional networks for invariant speech rep- resentations. InterSpeech, pp. 3229–3233, 2015.

[11] Worrall, D.E., Garbin, S.J., Turmukhambetov, D., Brostow, G.J.: Harmonic net- works: Deep translation and rotation equivariance. In: CVPR. (2017) 5028–5037

[12] Sohn, K., Lee, H.: Learning invariant representations with local transformations. In: CVPR, Omnipress (2012) 1339–1346

[13] Gens, R., Domingos, P.M.: Deep symmetry networks. In: Advances in neural
information processing systems. (2014) 2537–2545

[14] Sifre, L., Mallat, S.: Rotation, scaling and deformation invariant scattering for texture discrimination. In: CVPR, IEEE (2013) 1233–1240

[15] Henriques, J.F., Vedaldi, A.: Warped convolutions: Efficient invariance to spatial transformations. In: Int. Conf. on Machine Learning. (2017) 1461–1469

[16] Dieleman, S., De Fauw, J., Kavukcuoglu, K.: Exploiting cyclic symmetry in con-volutional neural networks. arXiv preprint arXiv:1602.02660 (2016)