Visual Localization using Capsule Networks

A Thesis Report

submitted by

OMKAR PATIL

ME15B123

in partial fulfilment of the requirements

for the award of Interdisciplinary Dual Degree with

BACHELOR OF TECHNOLOGY IN MECHANICAL ENGINEERING

MASTER OF TECHNOLOGY IN ROBOTICS

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

23rd June 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **Visual Localization using Capsule Networks**, submitted by **Omkar Patil**, to the Indian Institute of Technology, Madras, for the award of **Interdisciplinary Dual Degree**, is a bona fide record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Anurag Mittal
Research Guide
Professor
Dept. of Computer Science
IIT-Madras, 600 036

Place: Chennai

Date: 23rd June 2020

ACKNOWLEDGEMENTS

I would like to sincerely express my gratitude to Prof. Anurag Mittal, who as my project guide allowed me to work in the direction of my interest while supporting me with advice wherever needed. I am eternally grateful to IIT Madras for providing me with a nurturing environment to study, work and live in.

To my parents, I shall remain indebted always, for the constant emotional support which enables me to aim high and achieve my goals. I am forever thankful to my sister, Gowri - her cheerful attitude is a steady source of my encouragement. Lastly, I am thankful to my beagle 'Boltu', with whom I'm lucky to spend time, due to the COVID crisis.

Omkar Patil

ABSTRACT

KEYWORDS: Thesis; Capsules; Camera Pose Estimation; Structure from Motion

Camera pose estimation is a crucial task for many technologies which involve localization such as mobile robots and augmented reality. This task involves predicting the 6-DoF pose of the camera which has captured the given image from a known environment.

Presently, there are two thrust areas- the first and more successful- using geometry based computer vision techniques, and the second and emerging is utilization of deep learning models.

This project analyzed the current efforts made towards this task and proposes new methods to try and achieve competitive results. We explored the use of 'Capsules' for camera pose estimation. Capsule networks have better geometrical interpretability than CNNs, and have shown promising results on classification datasets like MNIST.

With Capsules, we achieved better results than with baseline PoseNet on small NORB dataset, modified for the task of camera pose estimation. The accuracy of the proposed network was found to be sub-par to that of PoseNet on the Shop Facade dataset. Feature visualizations for both the networks produced more insights on their performance and behaviour. We found that there is a scope for improvement and hence propose a few directions for future effort.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
CNN	Convolutional Neural Network
SFM	Structure from Motion
DOF	Degrees of Freedom
DR	Dynamic Routing
EM	Expectation Maximization
LSTM	Long Short-term Memory
IMU	Inertial Measurement Unit
GPS	Global Positioning System
SIFT	Scale Invariant Feature Transform

CHAPTER 1

INTRODUCTION

1.1 Camera Pose Estimation

Robot localization is knowing the exact location and orientation of a robot and is crucial for mobile robots, navigation, and augmented reality. Visual localization enables autonomous vehicles to navigate their surroundings and enables augmented reality applications to link virtual with real world. However, for visual localization approaches to be practical, they need to be robust to a wide variety of viewing conditions, including day-night changes, as well as weather and seasonal variations, while providing highly accurate 6 Degrees-of-Freedom (6-DoF) camera pose estimates. This problem takes the form of SLAM - simultaneous localization and mapping. This project examines a subset of the bigger problem - if we have already mapped the environment using a camera, can we predict the pose of the robot given a camera frame at any instance?

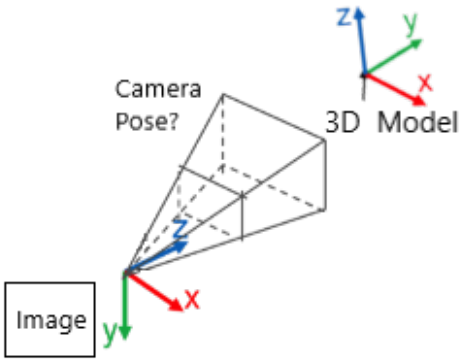


Figure 1.1: Predicting the 6-DoF pose of the camera which captured the given image from a known environment

Camera pose prediction has received widespread attention from the computer vision community in the form of Structure from Motion. SFM techniques which use descriptor matching have been fairly successful in camera pose prediction. Of late, the deep learning community has tried to tackle this problem with the introduction of end-to-end pose prediction algorithms using CNNs.

1.2 Literature Review

1.2.1 Classical Localization Approaches

There are two traditional ways to approach a local feature-based model for localization. First, the 'Place Recognition' method discretizes the world into a number of land-marks and attempts to identify which place is visible in a given image. This is modelled as an image retrieval problem. Sivic and Zisserman (2003) and Jégou *et al.* (2010) used image retrieval techniques like Bag-of-Words (BoW) and VLAD, respectively to identify the most similar image, based on a set of features from the geo-tagged database. The pose label of the most similar image is then taken as the pose for the queried image. According to Torii *et al.* (2015), DenseVLAD aggregates densely extracted SIFT descriptors, while NetVLAD uses learned features. Both are robust against day-night changes and work well at large-scale. However, all these networks must discretise the world into places and are unable to produce a fine grained estimate of 6-DoF pose.

Structure-based localisation techniques use a 3D model to represent the scene, usually created through Structure from Motion. Donoser and Schmalstieg (2014) determined the full 6-DoF camera pose of a query photo from a set of 2D-3D correspondences established via matching features found in the query, against descriptors associated with the 3D points. Further improvements to this have been made by Brachmann *et al.* (2017) to accelerate the algorithm and reduce its memory footprint. 3D coordinate regression methods establish the matches by regressing 3D coordinates from pixel patches. Sattler *et al.* (2019) reported that 3D coordinate regression methods currently achieve a higher pose accuracy at small scales. However, these methods require a 3D model with a large database of features and efficient retrieval methods. They are expensive to compute, often do not scale well, and are often not robust to changing environmental conditions.

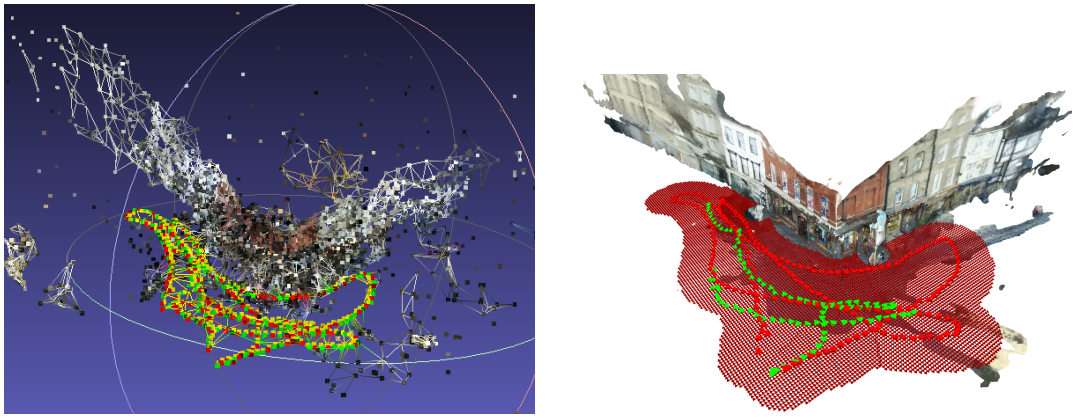


Figure 1.2: On the left- Shop Facade 3D points generated through Bundler, visualized on MeshLab (Cignoni *et al.* (2008)). This is later fed as an input to Active Search to generate pose estimates. On right- Visualization of the Shop Facade dataset generated by Sattler *et al.* (2019)

Sequence-based approaches for image retrieval are used for loop-closure detection in robotics. Milford and Wyeth (2012) produced impressive results with SeqSLAM, which required a matched sequence of images in the correct order. It significantly reduced false positive rates compared to single-image retrieval approaches.

The overall run-time of classical localization approach depends on the number of features found in a query image, the number of 3D points in the model, and the number of found correspondences and/or the percentage of correct matches. Moreover, SIFT-based methods do not work for datasets with repetitive structures and large, textureless regions. These drawbacks have prompted the machine learning community to tackle the problem with deep networks where the run-time scales with the size of the network and not of the scene.

1.2.2 Absolute Pose Regression Approach

Absolute camera Pose Regression (APR) train CNN to regress the camera pose of an input image. They all follow the same pipeline: Features are extracted using a base network, which are then embedded into a high-dimensional space. This embedding is then used to regress the camera pose in the scene. The current approaches mainly differ in the underlying base architecture and the loss function used for training.

Kendall *et al.* (2015) first proposed PoseNet, to directly regress 6-DoF camera pose from an input image with GoogLeNet. Kendall and Cipolla (2017) then extended

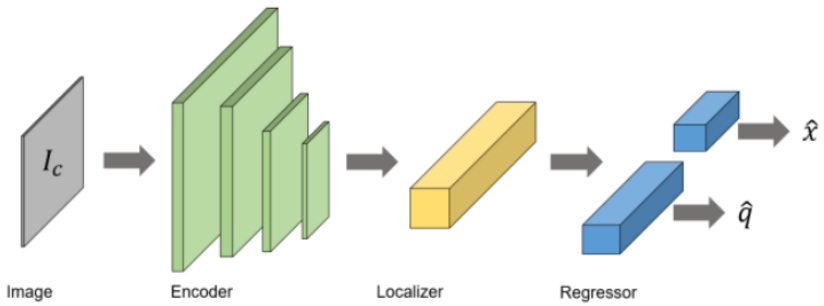


Figure 1.3: Architecture of PoseNet, which uses an end-to-end model for regressing the camera pose

Shavit and Ferens (2019)

PoseNet by learning the weight between camera translation and rotation loss and incorporated the reprojection loss. Melekhov *et al.* (2017) improved PoseNet with skip connections with ResNet34 architecture. Walch *et al.* (2016) used LSTM for dimensionality reduction on the image encoding with the assumption that high-dimensionality of the image encoding (when compared to the relatively small number of training examples), resulted in overfitting. Brahmbhatt *et al.* (2018) used visual odometry constraint between a pair of images, translation constraint from two GPS readings, and rotation constraint from two IMU readings to make the learning 'geometry-aware'.

Walch *et al.* (2016) have demonstrated PoseNet's efficacy on featureless indoor environments, where they show that SIFT based SFM techniques fail in the same environment.

Figure 1.4: Saliency maps suggest that convnets exploit not only distinctive points but also textureless patches.

Kendall* et al. *(2015)

However, PoseNet and other end-to-end CNN based approaches fail to produce accurate pose estimates. Sattler *et al.* (2019) showed that APR methods do not learn the geometry of the scene, but merely interpolate the pose within the range of the training dataset. They conducted an experiment to validate their theory where APR based approaches failed to extrapolate beyond a straight line on which they were trained. Results showed

that APR methods were significantly less accurate than structure-based methods and did not consistently outperform a handcrafted image retrieval baseline.

Table 1.1: Result of different models on Shop Facade dataset

$Model$	$Position/OrientationError$
PoseNet	1.19 / 6.88
MapNet	1.07 / 4.70
Active Search	**0.01 / 0.04**
DenseVLAD	0.98 / 7.90

CHAPTER 2

THEORY OF CAPSULE NETWORKS

2.1 Translational Invariance of CNNs

Features generated from CNNs are intentionally made invariant of the object's position through pooling operations, to improve object detection capabilities. Formally, a function is equivariant if applying any transformation to the input of the function has the same effect as applying that transformation to the output of the function. Invariance is a related notion, and is implied when the function applying transformations to the input does not change the output. Parameter sharing across spatial locations in CNNs make them translationally equivariant, which helps in learning and generalization. This is excluding the max pooling operation, which makes the network invariant to small translations.

Jaderberg *et al.* (2015) introduced spatial transformer module which disentangled the object pose from texture and shape, making the network invariant to translation, scale, rotation and more generic warping. They modelled the transformation required for each image into its canonical form which achieves the least classification error, leading to state of the art performance on several benchmarks like distorted MNIST.

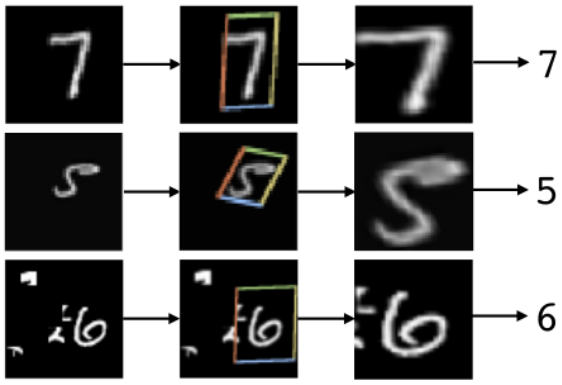


Figure 2.1: Effect of CNN with spatial transformer on distorted MNIST

Jaderberg* et al. *(2015)

Modelling was done here to finally transform the image into a canonical form which yields the least classification error. This affine invariance works against understanding the geometry of the scene, where object pose is a crucial aspect.

The objective in camera pose estimation is to predict pose (position and orientation) of the camera which clicked the given image, from a known surrounding. In other words, we have a dataset of images with pose of the camera that clicked each image. When given a new image, we would like to predict pose of the camera which clicked it. The images clicked by a camera are projectively related if the camera centre is same, or the image points lie on a plane. Naturally, we could expect better results if our model is not invariant to affine degrees of freedom.

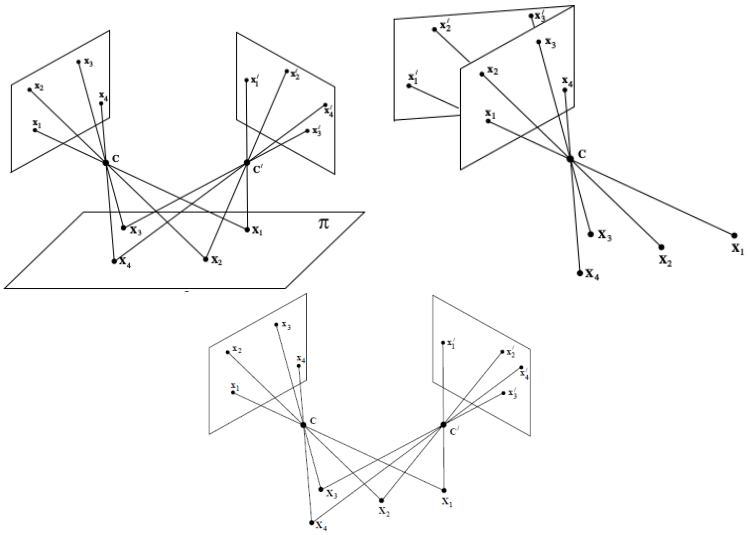


Figure 2.2: The images in top two scenarios are projectively related, but are not related by a homography in the bottom one.

Hartley and Zisserman (2003)

It would be very useful for the task at hand - pose prediction, if convolutional neural networks could be made equivariant to other affine degrees of freedom like rotation, shear and scale. It is tempting to exploit other effects of viewpoint changes by replicating features across scale, orientation and other affine degrees of freedom. Cohen and Welling (2016) proposed group-equivariant convolutional networks (G-CNN), which are equivariant to the action of a chosen group G. But this quickly leads to cumbersome high-dimensional feature maps that are not easy to achieve, either. This made us to look for a fundamentally different way in which neural networks are organized - Capsules.

2.2 Capsules to the Rescue

A capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or an object part. The activities of the neurons within an active capsule can represent the various properties including many different types of instantiation parameters such as pose (position and orientation), deformation, hue, texture, etc. Objects and object parts are abstract entities and need not conform to any physical interpretation. Neither can we pin-point what instantiation parameters a capsule will represent, beforehand. In most networks, object parts are derived from the features detected by a preceding convolutional neural network. Objects could be considered to be a set of parts with agreeing instantiation parameters put together.

For the purpose of camera pose estimation, the network will have to predict pose of the camera which clicked the input image. The pose (position and orientation) of objects in an image is related to pose of the camera which clicked the image. The lowest level capsules in the network convert pixel intensities into part instantiation parameters and then the further layers go up the hierarchy by multiplying the output of the previous layer with the appropriate part-object transformation matrix. This implies that if the camera image undergoes a transformation, the instantiation parameters will change accordingly and so will the output pose. This property of capsules lends a better geometrical interpretation to the network.

2.3 Theory of Capsules

Hinton *et al.* (2011) first released the idea of the capsules in the form of auto-encoders in 2011, where a network of capsules was used to decode images into an affine-equivariant form and the probabilistically mix the decoder reconstructions to obtain the affine transformed image. The intuition behind capsules was that each capsule will output a pose of the part it is representing in the image. This pose output will be multiplied with a set of weights and then fed into the next layer of capsules, whose activations represent objects composed of several parts. The set of weights here, represent the part object transformation, and the part poses which align together to produce the maximum probability of

presence of a specific object will be given higher weights.

An interesting property of this method of shape recognition is that the knowledge of part-whole relationships is viewpoint-invariant and is represented by weight matrices, whereas the knowledge of the instantiation parameters of currently observed objects and their parts is viewpoint-equivariant and is represented by neural activities. However, this paper did not delve into the mechanism with which this could be achieved, and carried forward with only one layer of capsules to reconstruct the transformed images from MNIST. The network was also trained on changing viewpoints on 3D objects, and produced very good reconstructions of objects from a new viewpoint.

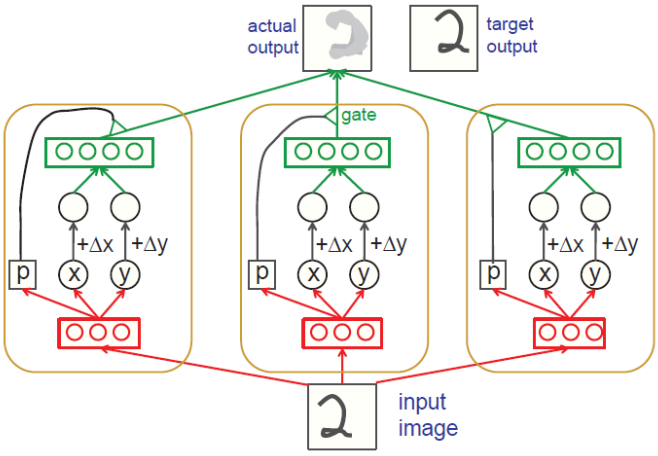


Figure 2.3: Capsules of a transforming auto-encoder that models translations. Each capsule in the figure has 3 recognition units and 4 generation units.

Hinton* et al. *(2011)

2.3.1 Dynamic Routing Capsules

Sabour *et al.* (2017) proposed a multi-layer capsule network in 2017 (DRCapsNet), which produced state of the art performance on MNIST. First, a convolutional layer was used to extract the basic features from the image, followed by a primary layer of thirty-two 8D capsules. This was followed by ten 16D capsules which represented the various instantiation parameters of the ten digits. The network used the length of a capsule vector as a proxy for presence probability of an object, and hence the capsule having the longest vector length decided the classification of the image. Two types of losses were used to train the image- L2 loss and reconstruction loss, between the expected image and the reconstructed image of the predicted digit from its instantiation

parameters.

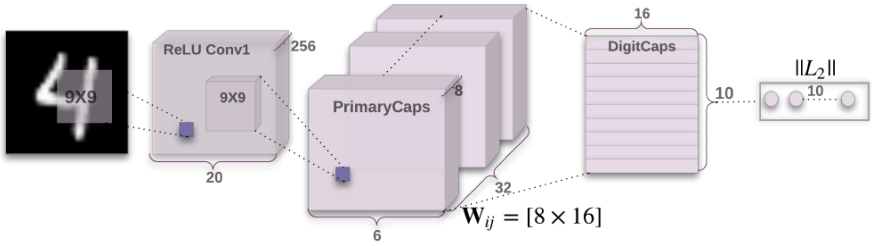


Figure 2.4: DRCapsNet architecture: Note that the diagrams represent feature size of the outputs of each layer.

Sabour* et al. *(2017)

DRCapsNet utilized dynamic routing for solving the problem of assigning parts to wholes- that is to instantiate larger parts up the hierarchy by using instantiation parameters of capsules in the previous layer and the part-whole relationships. Finding tight clusters of high-dimensional votes that agree in the midst of irrelevant votes is non-trivial because we cannot grid the high-dimensional pose space in the way the low-dimensional translation space is gridded to facilitate convolutions. To solve this challenge, the authors used a fast iterative process called 'routing- by-agreement' that updates the probability with which a part is assigned to a whole based on the proximity of the vote coming from that part, to the votes coming from other parts that are assigned to that whole.

Dynamic routing is done by weighing all the inputs to a capsule j, $u_{j|i}$ by a coupling coefficient c_{ij}. For each possible parent, the capsule computes a prediction vector by multiplying its own output u_i by a weight matrix W_{ij}. If this prediction vector has a large scalar product with the output of a possible parent, there is top-down feedback which increases the coupling coefficient for that parent and decreases it for other parents. This increases the contribution that the capsule makes to that parent, and thus further increasing the scalar product of the capsule's prediction with the parent's output.

Procedure 1 Routing algorithm.

procedure ROUTING($\hat{\boldsymbol{u}}_{j|i}$, r, l)
  for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
  for r iterations **do**
    for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$
    for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
    for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$
    for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
  return $\mathbf{v}_j$

Figure 2.5: Dynamic Routing Algorithm

Sabour* et al. *(2017)

The squash function is a non linearity introduced to ensure that short vectors get shrunk to almost zero length and long vectors get shrunk to a length slightly below 1. Note that, here vectors represent the presence probability of the object the capsule represents. The squashed output is:

$$v_{\mathrm{j}} = \frac{\|s_{\mathrm{j}}\|^2}{1 + \|s_{\mathrm{j}}\|^2} \frac{s_{\mathrm{j}}}{\|s_{\mathrm{j}}\|}$$

2.3.2 Matrix Capsules with EM Routing

The next paper in series by Hinton *et al.* (2018) described a version of capsules in which each capsule has a logistic unit to represent the activation of an entity and a 4x4 matrix which could learn to represent the relationship between that entity and the viewer (the pose). There are two improvements in EMCapsNet over DRCapsNet. Firstly, a capsule in DRCapsNet consists of a vector of instantiation parameters, whereas a capsule in EMCapsNet forms a 4x4 matrix. The presence probability of a 'part' in DRCapsNet is the length of the capsule vector representing that part. EMCapsNet reserves a separate unit for every capsule to represent its activation. The second improvement is the method by which 'parts' are assigned to 'wholes', or in other words the way by which activations in one layer are used by the next. DRCapsNet uses dynamic routing to assign parts to wholes. EMCapsNet uses Expectation-Maximization for the same.

A capsule in one layer votes for the pose matrix of many different capsules in the layer above by multiplying its own pose matrix by trainable viewpoint-invariant transformation matrices that could learn to represent part-whole relationships. Each of these votes is weighted by an assignment coefficient. These coefficients are iteratively updated for

each image using the Expectation-Maximization algorithm such that the output of each capsule is routed to a capsule in the layer above that receives a cluster of similar votes. The transformation matrices are trained discriminatively by backpropagating through the unrolled iterations of EM between each pair of adjacent capsule layers.

DRCapsNet with dynamic routing achieved 2.7% test error rate on small NORB, which was on-par with the state-of-the-art. EMCapsNet with expectation maximization routing was successful in reducing the test error rate to 1.8%

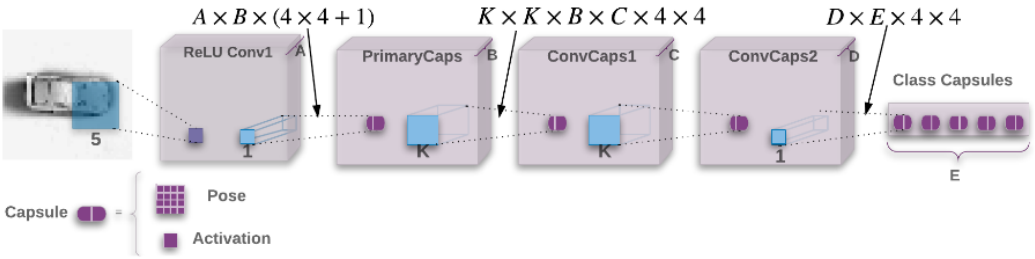


Figure 2.6: EMCapsNet Architecture : A network with one ReLU convolutional layer followed by a primary convolutional capsule layer and two more convolutional capsule layers. Here, A, B, C, D, E, K are network hyperparameters.

Hinton* et al. *(2018)

As Hui (2017) explained, in EM clustering, we represent datapoints by a Gaussian distribution. In EM routing, we model the pose matrix of the parent capsule with a Gaussian also. The pose matrix is a 4x4 matrix, i.e. 16 components. We model the pose matrix with a Gaussian having 16 μ and 16 σ and each μ represents a pose matrix component.

The pose matrix and the activation of the output capsules are computed iteratively using the EM routing. The EM method fits datapoints into a mixture of Gaussian models with alternative calls between an E-step and an M-step. The E-step determines the assignment probability r_{ij} of each datapoint to a parent capsule. The M-step re-calculates the Gaussian models' values based on r_{ij}. We repeat the iteration 3 times. The last a_j will be the parent capsule's output. The 16 μ from the last Gaussian model will be reshaped to form the 4x4 pose matrix of the parent capsule.

We initialize the assignment probability r_{ij} to be uniformly distributed. i.e. we start with the children capsules equally related with any parents. We call M-step to compute an

updated Gaussian model (μ, σ) and the parent activation a_j from a, V and current r_{ij}. If the benefit β_a of representing datapoints by the parent capsule *j* outranks the cost caused by the discrepancy in their votes, we activate the output capsules. We do not compute β_a analytically. Instead, we approximate it through training using the backpropagation and a cost function. Then we call E-step to recompute the assignment probabilities r_{ij} based on the new Gaussian model and the new a_j.

Procedure 1 Routing algorithm returns **activation** and **pose** of the capsules in layer $L+1$ given the **activations** and **votes** of capsules in layer L. V_{ij}^h is the h^{th} dimension of the vote from capsule i with activation a_i in layer L to capsule j in layer $L+1$. β_a, β_u are learned discriminatively and the inverse temperature λ increases at each iteration with a fixed schedule.

procedure EM ROUTING($\boldsymbol{a}, V$)
  $\forall i \in \Omega_L, j \in \Omega_{L+1}$: $R_{ij} \leftarrow 1/|\Omega_{L+1}|$
  for t iterations **do**
    $\forall j \in \Omega_{L+1}$: M-STEP($\boldsymbol{a}$, R, V, j)
    $\forall i \in \Omega_L$: E-STEP(μ, σ, $\boldsymbol{a}$, V, i)
  return $\boldsymbol{a}, M$

procedure M-STEP($\boldsymbol{a}$, R, V, j) ▷ for one higher-level capsule, j
  $\forall i \in \Omega_L$: $R_{ij} \leftarrow R_{ij} * \boldsymbol{a}_i$
  $\forall h$: $\mu_j^h \leftarrow \frac{\sum_i R_{ij} V_{ij}^h}{\sum_i R_{ij}}$
  $\forall h$: $(\sigma_j^h)^2 \leftarrow \frac{\sum_i R_{ij} (V_{ij}^h - \mu_j^h)^2}{\sum_i R_{ij}}$
  $cost^h \leftarrow \left(\beta_u + log(\sigma_j^h)\right) \sum_i R_{ij}$
  $a_j \leftarrow logistic(\lambda(\beta_a - \sum_h cost^h))$

procedure E-STEP(μ, σ, $\boldsymbol{a}$, V, i) ▷ for one lower-level capsule, i
  $\forall j \in \Omega_{L+1}$: $\boldsymbol{p}_j \leftarrow \frac{1}{\sqrt{\prod_h^H 2\pi(\sigma_j^h)^2}} \exp\left(-\sum_h^H \frac{(V_{ij}^h - \boldsymbol{\mu}_j^h)^2}{2(\sigma_j^h)^2}\right)$
  $\forall j \in \Omega_{L+1}$: $\boldsymbol{R}_{ij} \leftarrow \frac{\boldsymbol{a}_j p_j}{\sum_{k \in \Omega_{L+1}} \boldsymbol{a}_k p_k}$

Figure 2.7: Expectation Maximization Algorithm for EMCapsNet

Hinton* et al. *(2018)

CHAPTER 3

THE POSECAP MODEL

3.1 Model Architecture

We created a new model called 'PoseCap' based on dynamic routing capsules for the task of camera pose estimation. A convolutional layer was used to extract the basic features from the image, followed by a primary capsule layer to generate 7D (vector) capsules. These capsules were multiplied by a weight matrix and added in ratio of their coupling coefficients to produce the final 7D capsule. This 7D capsule represents the pose of the camera- the first 3 values being the location and last 4 the orientation in quarternion form.

The primary capsule layer is equivalent to seven structurally similar convolutional layers clubbed together to produce a 7D output from every convolution. The 8192 -7D capsule outputs are then multiplied with weights representing part-whole transformations. Three iterations of dynamic routing are performed on these capsule outputs for every input image to produce the camera pose estimation.

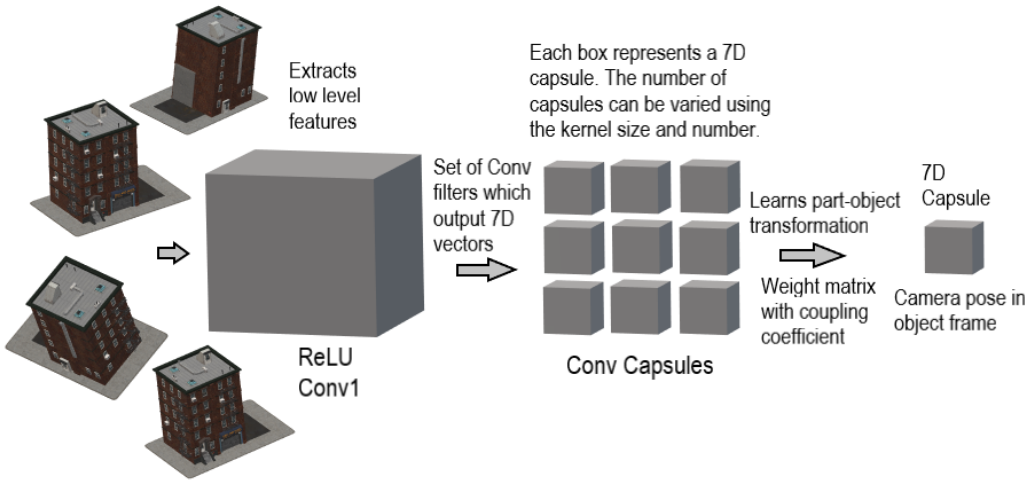


Figure 3.1: Posecap Network Architecture

Table 3.1: Model Architecture Details

$LayerName$	$Details$	$Capsules$
Conv	Conv2d: filters = 256, kernel size = 9, stride= 1	0
Primary Capsules	7x[Conv2d:filters=32, kernel size=9, stride=2]	8192
Pose Capsule	FC with dynamic routing : 8192 x 7	1

3.2 Model Hypothesis

Let us consider that a car will appear differently when looked at from different positions and orientations. When the appearance of a car changes due to change in observational pose; the set of lines, shapes, curves and colours of the car seen, also change. This set of attributes (lines and colours) together confined to a small patch in an image (whose size is decided by the filter dimensions) is what is referred to as a 'part'.

When an image is fed into this network, the convolutional layer begins to extract the elementary features from it. Part instantiation parameters are then learnt from these features in the primary capsule layer, in over 8192 capsules. Instantiation parameters could be pose (position and orientation), deformation, hue and texture among others, describing that patch. Depending upon appearance of the car, only some features, and consequently only some capsules will get activated. Subsequently, all the instantiation parameters are multiplied by weights representing part-whole transformations. These weights are learnt while training, but are fixed irrespective of the camera position.

A 'whole' is considered to be a larger patch of texture, colour and shapes. A 'whole' is composed of many parts in specific poses. Thus the part-whole relationship is dependent on the scene, rather than the camera pose. Every part makes prediction for a whole, but all the wholes don't align. Dynamic routing is used to get clusters of parts whose wholes align, and the corresponding wholes as the output. The deeper we go with the network, larger the capsule representations in size.

We use the fact that the camera pose will be a function of how objects appear in its image. Hence, we assume camera pose to be a non-linear function of the instantiation parameters of the wholes, and route the primary capsule outputs to produce a camera pose estimate.

3.3 Loss Function

To train the model, we used the following loss function:

$$loss(I) = \|\hat{x} - x\| + \beta \left\| \hat{q} - \frac{q}{\|q\|} \right\|$$

where pose p is:

$$p = [x, q]$$

x being the position and q quarternion of the camera orientation.

Here beta is a hyperparameter, used to keep the expected value of position and orientation errors approximately equal. We tested extensively to find the optimal value of beta for our model.

3.4 Experimentation

3.4.1 Datasets

There are many published datasets for camera pose estimation, some of which also incorporate day-night and seasonal changes. The notable among them are 7-scenes dataset for indoor visual localization and the Cambridge Landmarks dataset for outdoor localization. Here we used the 'Shop Facade' scene from Cambridge Landmarks dataset (Kendall *et al.* (2015)) for initial experiments.

Figure 3.2: Shop Facade scene from Cambridge Landmarks

Kendall* et al. *(2015)

Sabour *et al.* (2017) discussed a drawback of Capsules which it shares with generative models that it likes to account for everything in the image. They attributed their model's poor performance to varied backgrounds in CIFAR-10, among other reasons. Similarly, we can expect the same for all the visual localization datasets, since they contain a heavy amount of noise such as pedestrians, vehicles and day-night changes. So as a pilot, we experimented our network on the small-NORB dataset (LeCun *et al.* (2004)).

This had multi-fold benefits - firstly, capsules have been able to achieve state of the art results on classification task of small NORB. Here, as we were interested in camera pose estimation, rather than predicting the class of the object given an image, we modelled the prediction of elevation and azimuth of the camera. This information and more on lighting can be found in the metadata of the dataset. Secondly, we saved precious computing power and time since this dataset is much lighter than the conventional datasets used for visual localization.

The small-NORB dataset contains images of 50 toys belonging to 5 generic categories: four-legged animals, human figures, airplanes, trucks, and cars. The objects were imaged by two cameras under 6 lighting conditions, 9 elevations (30 to 70 degrees every 5 degrees), and 18 azimuths (0 to 340 every 20 degrees). We modified the conventional data-input pipeline to output the azimuth and elevation given an image. This did not serve the full purpose of localization, as it only outputs the partial location information and no orientation information. But it served as a good starting point to see, if capsules can outperform CNN based networks in a relatively simple setting.

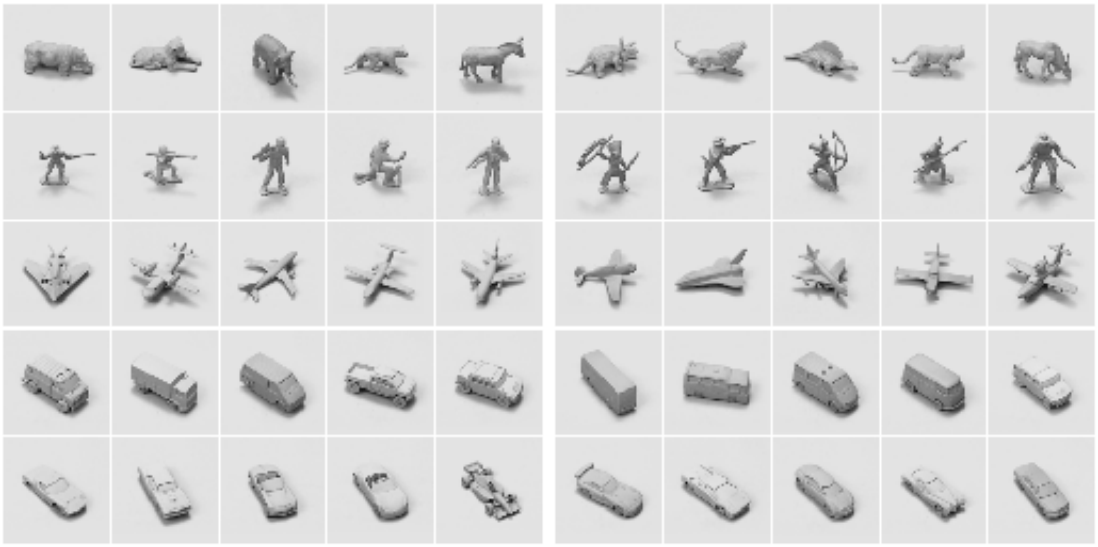


Figure 3.3: The small-NORB dataset. There are a total of 50 objects (5 categories in rows with each of their 10 instances).

LeCun* et al. *(2004)

There are a total of 50 objects in the small-NORB dataset, and each object has over 972 images taken from different camera orientations, from 2 different cameras. We considered all images from a single camera of the first object instance- a hippopotamus, for testing our model.

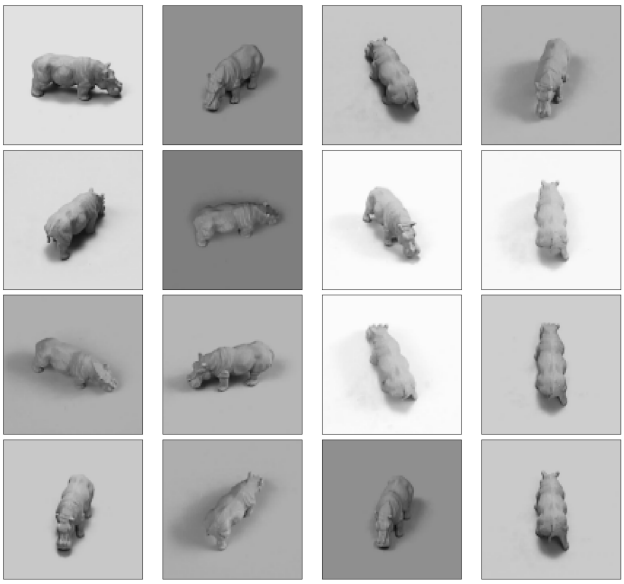


Figure 3.4: Object picked for training/testing the model.

We shuffled the dataset consisting of all the images of our chosen object using a seed of 420 on NumPy. This testing should be carried out on multiple seeds and the average results presented. We have shown the results only for one seed here. The table below shows the number of training and testing images in both the datasets.

Table 3.2: Number of training and testing images in both the datasets

$Dataset$	$TrainingImages$	$TestingImages$	$ImageSize$
small NORB	648	324	96x96
Shop Facade	231	103	455x256

3.4.2 Hyperparameters

We tested extensively to find the optimal hyperparameters for PoseCap and PoseNet on both the datasets- small NORB and Shop Facade. The hyperparameters are mentioned alongside the results. Adam optimizer was primarily used to train both the models. PoseCap did not train well with stochastic gradient descent with Nestorov momentum. The abbreviations used for hyperparameters, and their standard values are as follows:

Table 3.3: Abbreviations for Hyperparameters

$Abbreviations$	$Explanation$	$StandardValues$
beta1 and beta 2	Coefficients used for computing running averages of gradient and its square	0.9 and 0.999
batchSize	Number of training images used in a batch	54 for small NORB and 77 for Shop Facade
loadSize	Resize image to this size	256 for PoseNet and 56 for PoseCap
fineSize	Crop image to this size	224 for PoseNet and 48 for PoseCap
lr	Learning rate	0.01/ 0.001/ 0.0001
beta	Factor with which error in orientation is multiplied	1/ 100/ 500/ 1000

We used the above values for training, except where explicitly mentioned.

3.4.3 Baseline

We used PoseNet (Kendall *et al.* (2015)) as our baseline. PoseNet uses a modified version of GoogLeNet, a 22 layer convolutional network with six inception modules. The major modifications are addition of a fully connected layer before the final regressor of feature size 2048, and replacement of the 3 softmax classifiers with affine regressors.

The input image was rescaled so that the smallest dimension is 256 pixels before cropping to 224x224 pixel input to the GoogLeNet convnet.

CHAPTER 4

RESULTS

4.1 Results

As mentioned in the previous chapter, we tested PoseCap on two datasets - small NORB and Shop Facade. We used PoseNet as our baseline for comparison. The required output for small NORB dataset is elevation and azimuth of the camera which clicked the input image. The required output for Shop Facade dataset is the position and orientation of the camera. We modified the networks to suit the outputs accordingly. Other modifications to the networks are mentioned at appropriate places. The results for small NORB are showcased first, followed by results for Shop Facade.

4.1.1 Small NORB

Baseline CNN - PoseNet

The network was modified to output elevation and azimuth of the camera which clicked the given image. Posenet was trained for 3 values of learning rate for Adam optimization. The comparison of the three learning rates is shown in the graph and table below. It is evident from the graphs that lower learning rates achieved better results for PoseNet on small NORB.

Table 4.1: Testing results for different values of learning rate for PoseNet on small NORB

$LearningRate$	$MSELoss$
0.0001	**44.67**
0.0005	58.47
0.001	84.78

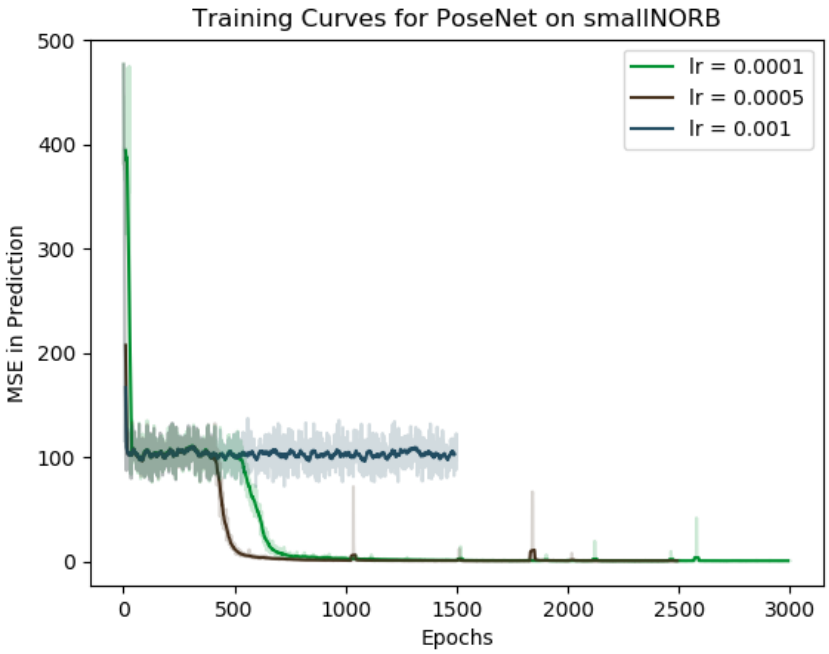


Figure 4.1: Training graph for PoseNet on small NORB - comparison on different learning rates

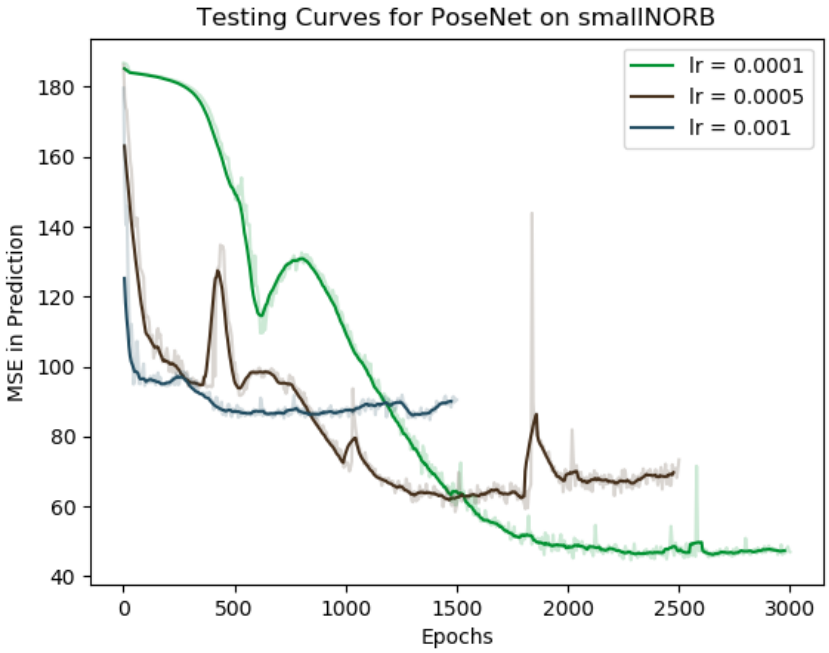


Figure 4.2: Testing graph for PoseNet on small NORB - comparison on different learning rates

PoseCap

Dynamic routing squashes the capsule output restraining the network learning. Squashing is a non-linear function which maps the output from 0 to 1 to turn capsule outputs into probability estimates. Hence the network did not learn as the outputs required for azimuth and elevation range from 0-360 and 0-90, respectively. This is one of the major deficiencies of dynamic routing based capsule networks.

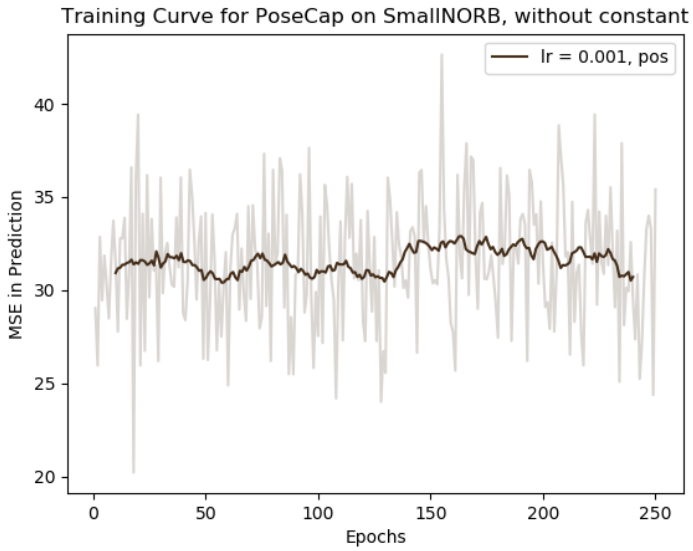


Figure 4.3: PoseCap on small NORB

To overcome this, we multiplied the capsule output by 360 and 90 for azimuth and elevation, respectively for the network to learn values between 0 and 1. This will not be possible in all cases, such as to predict position estimates, where the maximum value is not known. The table below gives the result and the graph shows the training and testing curves.

Table 4.2: Test result for PoseCap on small NORB

$LearningRate$	$Epochs$	$MSELoss$
0.001	1490	5.45

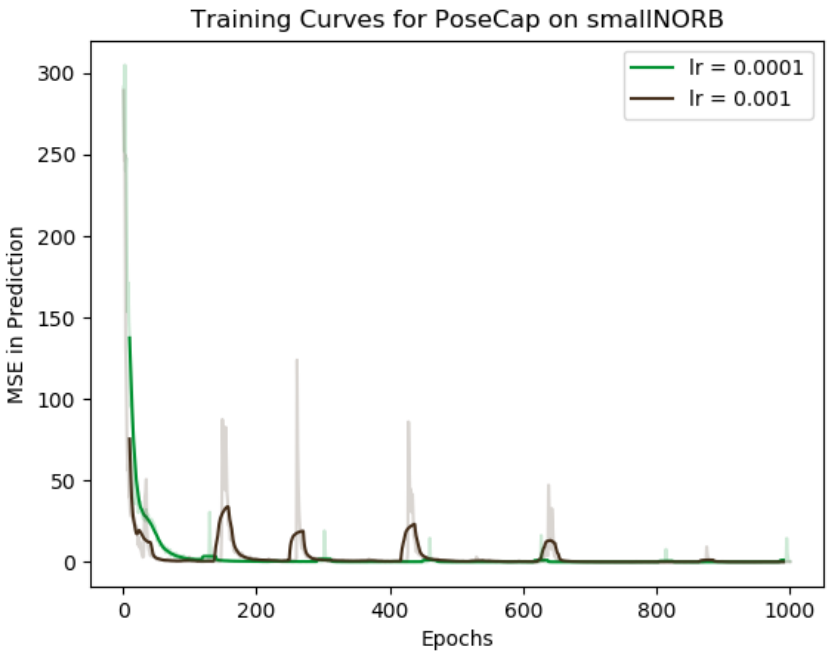


Figure 4.4: Training curve for PoseCap on small NORB

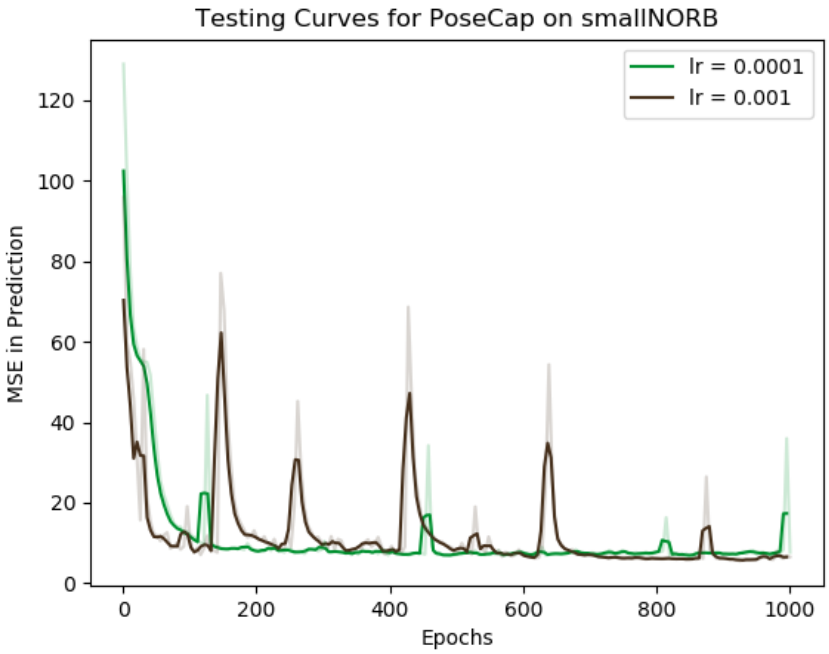


Figure 4.5: Testing curve for PoseCap on small NORB

Comparison between PoseCap and CNNs

As is evident from the results, PoseCap outperformed CNNs on small NORB by achieving a lower test error rate. The results are summarized in the table below. We rationalized this through feature visualizations, as given in the inferences section.

Table 4.3: Results on small NORB

PoseNet	*DRCapsNet*
44.67	**5.45**

4.1.2 Shop Facade

Baseline CNN - PoseNet

Posenet was trained for 3 values of learning rate with Adam optimization. The comparison of the three learning rates is shown in the graphs and table below.

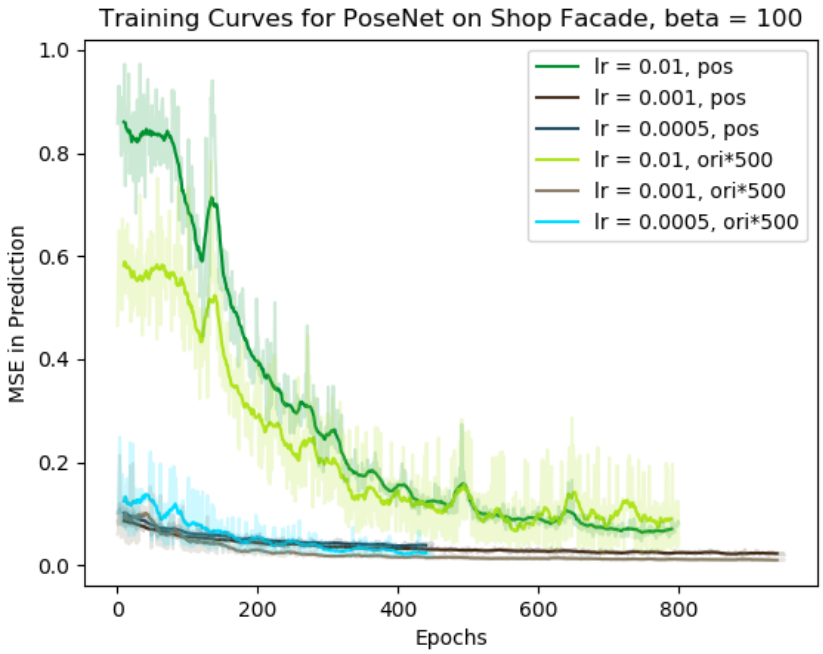


Figure 4.6: Training curves for PoseNet on Shop Facade

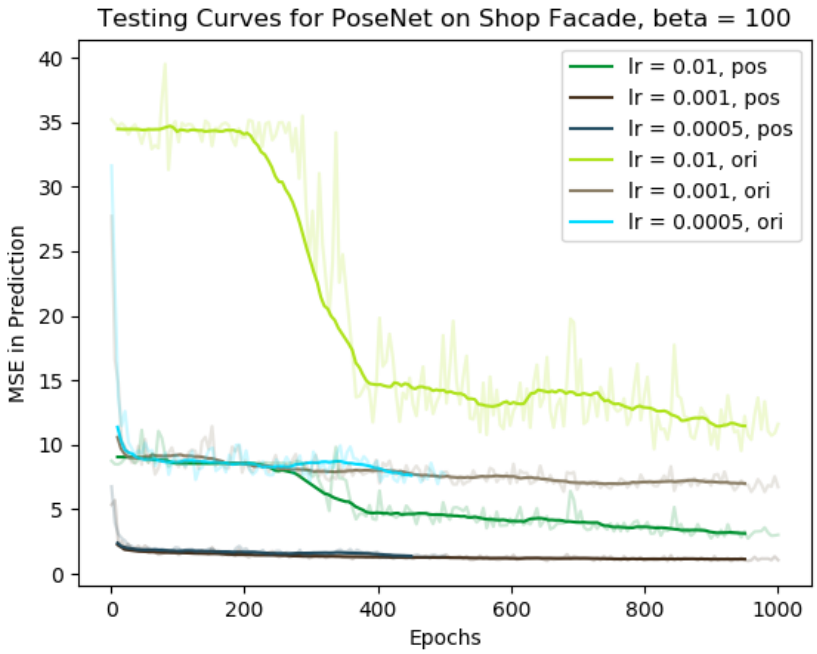


Figure 4.7: Testing curves for PoseNet on Shop Facade

Table 4.4: PoseNet error on Shop Facade

Position	*Orientation*	*Beta*	*lr*	*Epoch*
1.25m	7.33°	100	0.0005	425

PoseCap

Initially, we tested the PoseCap model without multiplying the outputs by a constant. We noticed that the position error did not decrease as the training progressed. This may be due to position outputs being greater than one. As discussed for small NORB, capsule networks based on dynamic routing have difficulty in learning values greater than one, as they squash the output.

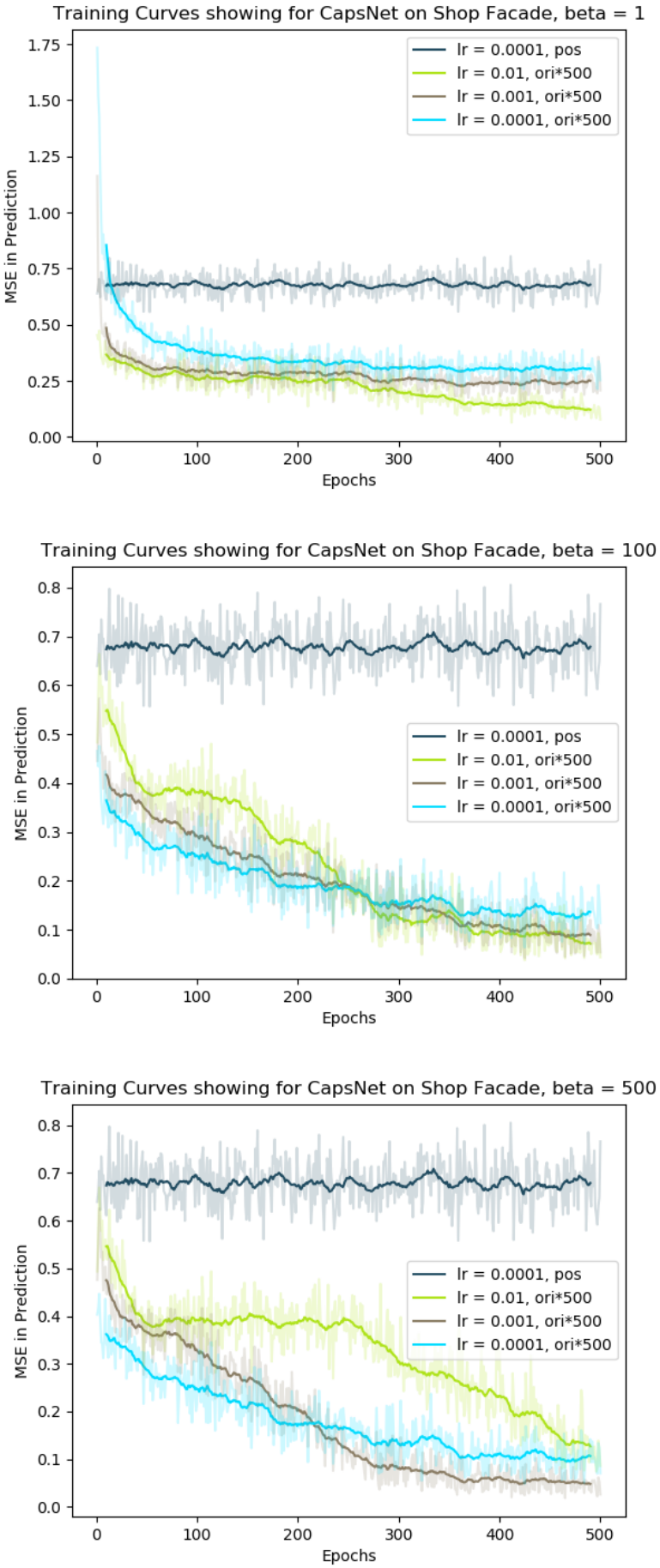


Figure 4.8: Training curves for PoseCap on Shop Facade

Thereafter, we trained for position and orientation separately. For the network learning position estimates, we multiplied the output with a constant - 100, which was greater than the maximum output expected. We observed that the network started to learn. The graphs and least error for position and orientation networks are given below.

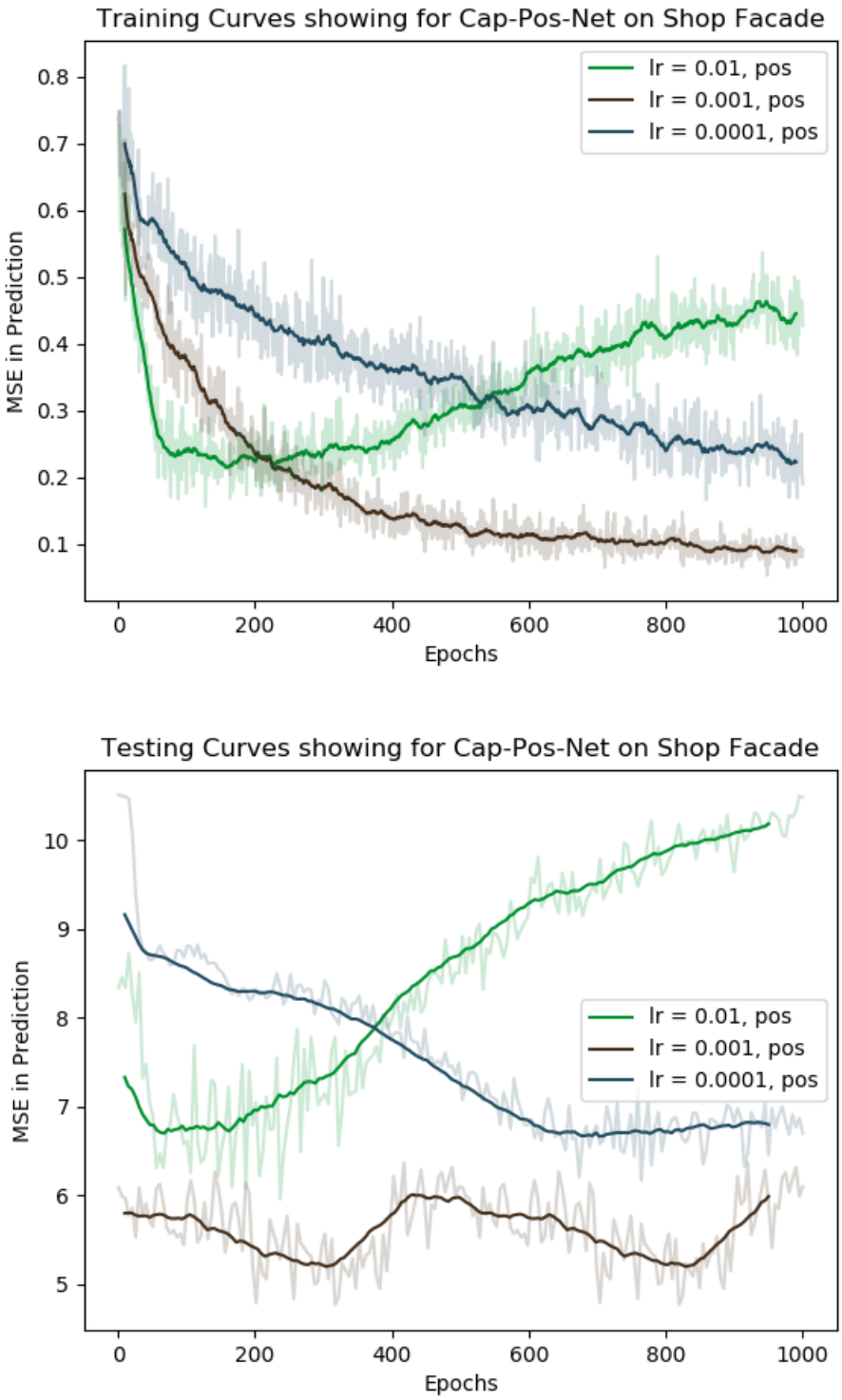
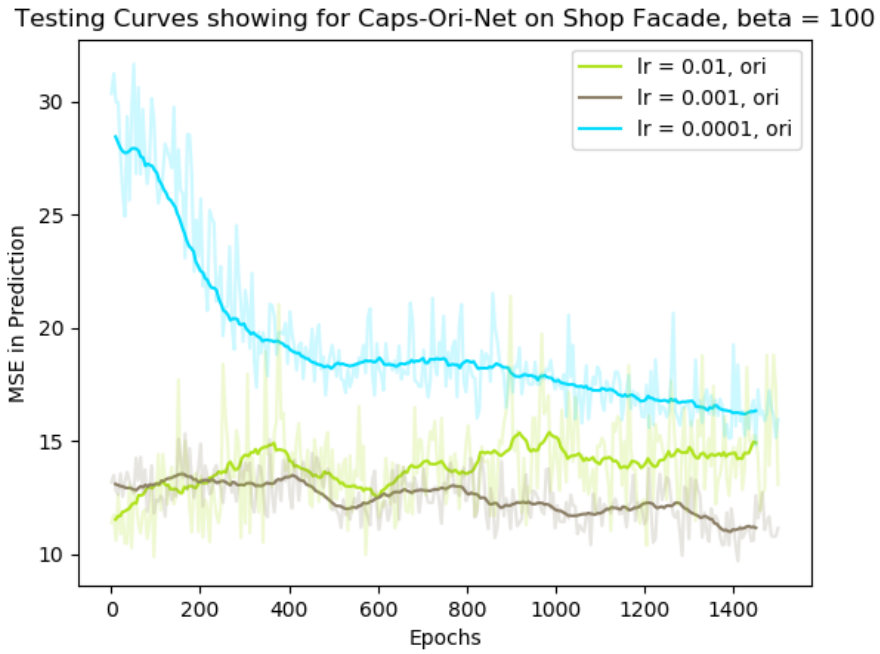


Figure 4.9: Training and Testing curves for Cap-Pos-Net on Shop Facade



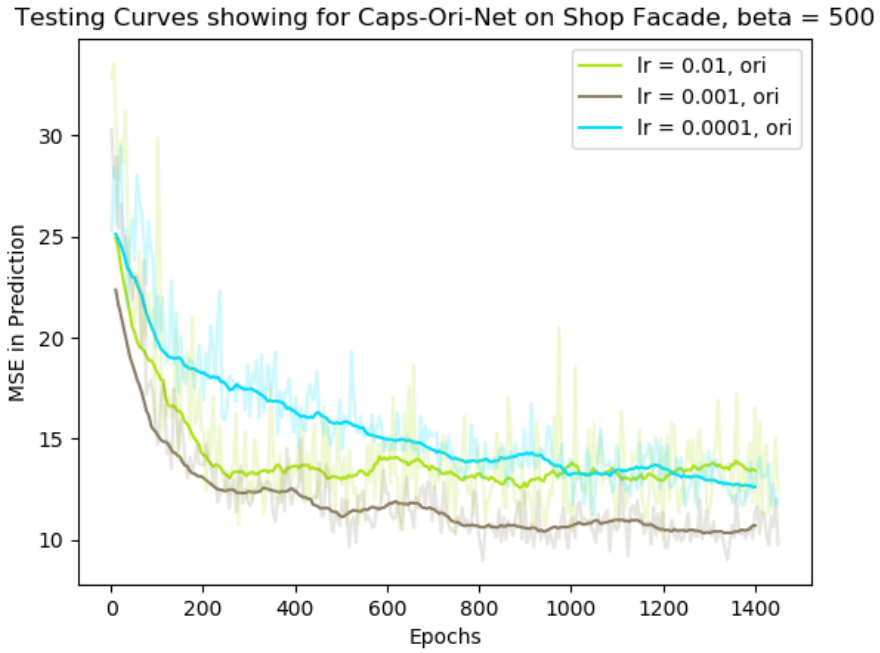


Figure 4.10: Testing curves for Cap-Ori-Net on Shop Facade

Table 4.5: Result of Individual Networks

Model	*Position*	*Orientation*
Min Error	4.77	5.45
Learning Rate	0.001	0.001
Beta	0	500
Epoch	320	860

Finally, we trained for both position and orientation in the same network, with the position outputs being multiplied by a constant - 100. The training curves for 4 values of beta are shown below.

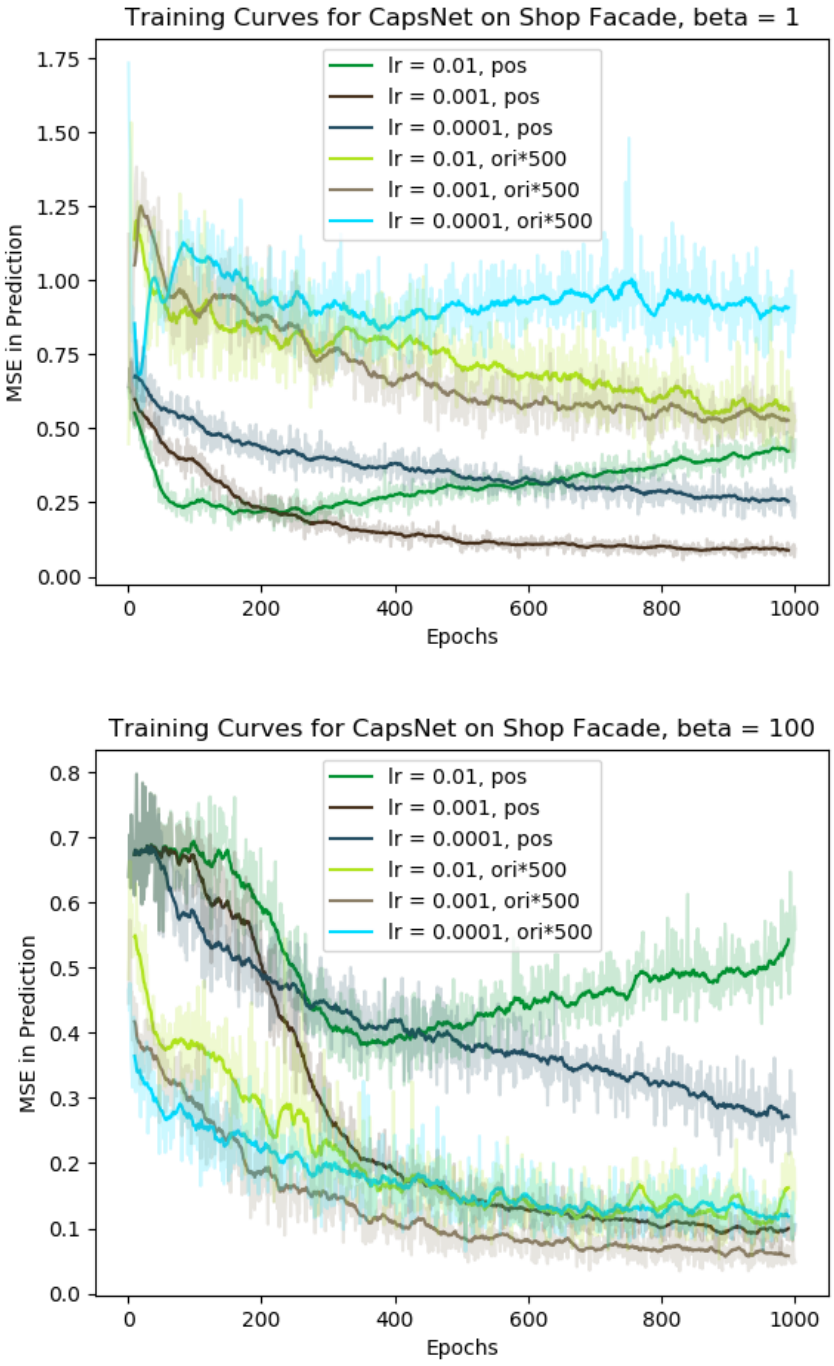
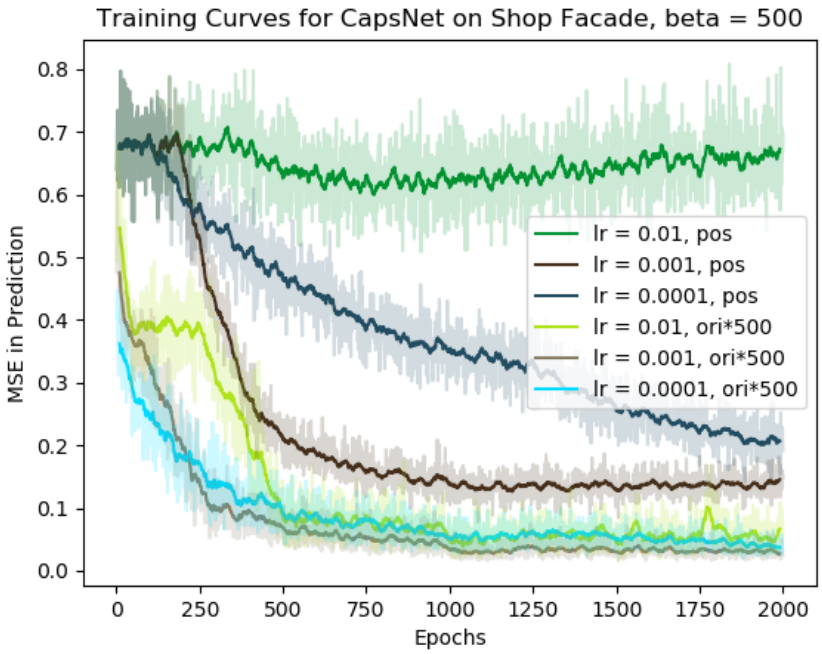


Figure 4.11: Training curves for PoseCap on Shop Facade, beta = 1, 100



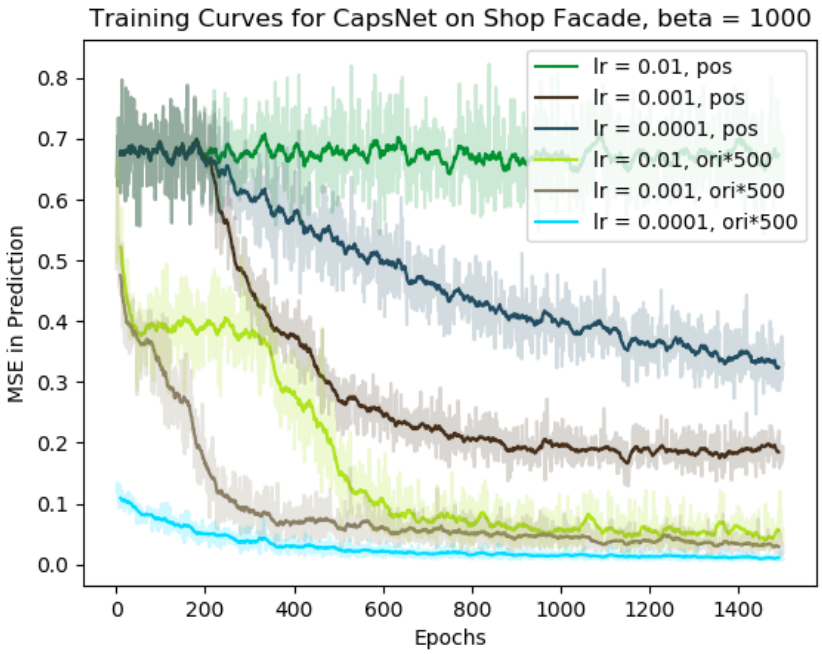


Figure 4.12: Training curves for PoseCap on Shop Facade, beta = 500, 1000

The testing curves for 4 values of beta are shown below along with a table for the least error obtained.

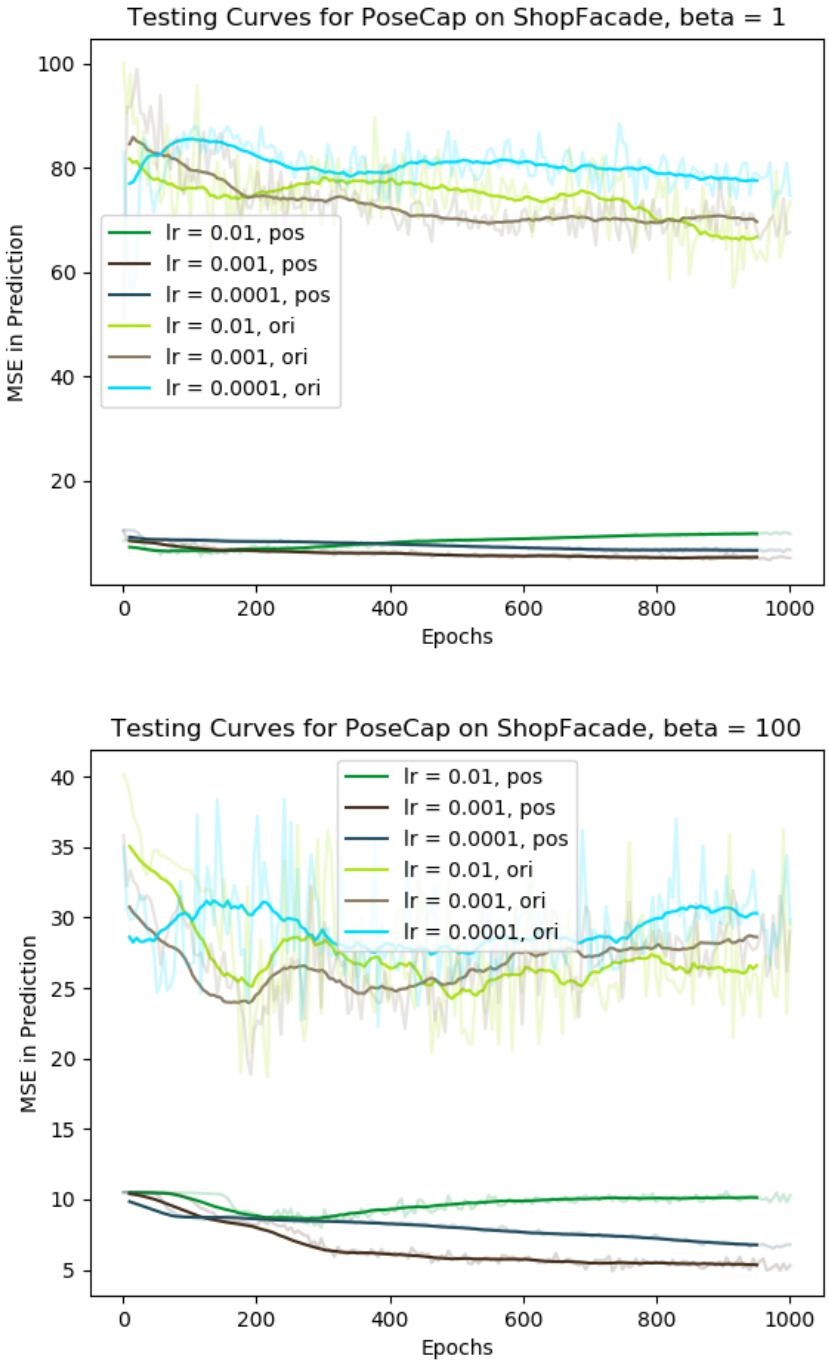
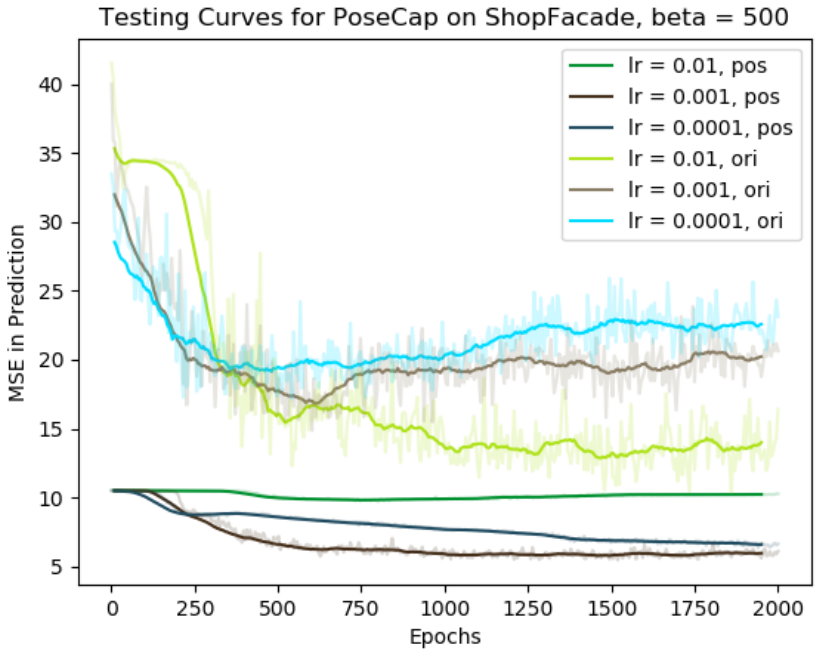


Figure 4.13: Testing curves for PoseCap on Shop Facade, beta = 1, 100



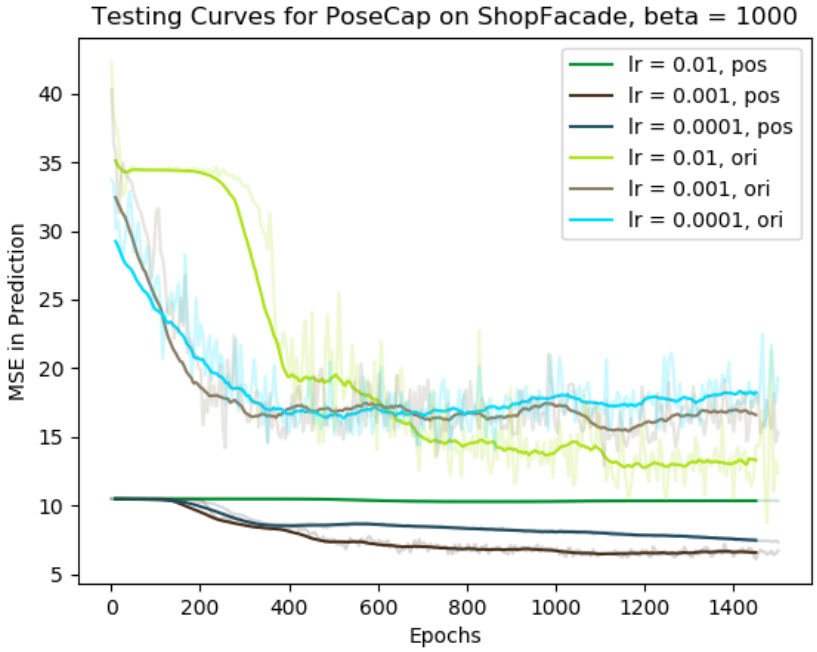


Figure 4.14: Testing curves for PoseCap on Shop Facade, beta = 500, 1000

Table 4.6: Result of PoseCap Network

Min Error	Pos: 6.49 | Ori: 12.91
Learning Rate	0.001
Beta	1000
Epoch	1110

Comparison between PoseCap and CNNs

PoseCap fell short in accuracy which was achieved by PoseNet, on Shop Facade. Individual networks for estimating camera position and orientation - Cap-Pos-Net and Cap-Ori-Net gave a better accuracy for orientation than PoseNet. These results look promising, given the fact that PoseCap model performed better on small NORB. Therefore, improved capsule-based networks may perform better than CNN- based networks on the task of camera pose prediction.

Table 4.7: Results on Shop Facade

Model	*Position Error*	*Orientation Error*
PoseNet	**1.25**	7.33
PoseCap	6.49	12.91
Caps-Ori-Net and Caps-Pos-Net	4.77	**5.45**

CHAPTER 5

INFERENCES AND FUTURE WORK

5.1 Inferences

We found that PoseCap (our proposed model based on Capsules) outperformed the chosen baseline- PoseNet, on small NORB dataset modified for the task of camera pose estimation. Each image in small NORB has only one non-occluded object in grayscale.

However, our model PoseCap performed sub-par to PoseNet on the Shop Facade dataset. A likely reason could be that we resized the original image from 455x256 to 48x48. In future, modifying the network to accept larger image sizes could lead to better performance. Sabour *et al.* (2017) reported that a possible drawback of Capsule based networks- in line with generative models- is that it tends to account for everything in the image. Shop Facade dataset has a high amount of noise including pedestrians in its images, which could explain PoseCap's unsatisfactory performance on it. Increasing the network size may consequently yield a rise in its pose prediction performance, making the model more robust to noise in images.

We delved into feature visualizations of both PoseCap and PoseNet for better understanding of the models. We followed the same nomenclature for layers as Szegedy *et al.* (2015) for PoseNet, and as given in table A.1 for PoseCap.

5.1.1 Feature Visualizations

Feature visualizations were generated by optimizing the input image which resulted in the maximum activation of the selected filter in a layer. We used the fact that neural networks are differentiable with respect to the input, and iteratively tuned the input inorder to minimize the loss function- which is the negative of the norm of feature activations.

Olah *et al.* (2017) showed how to remove high frequency patterns from the feature visualizations using regularization. However, we performed a comparative analysis between visualizations obtained from both the networks for two different datasets.

Small NORB

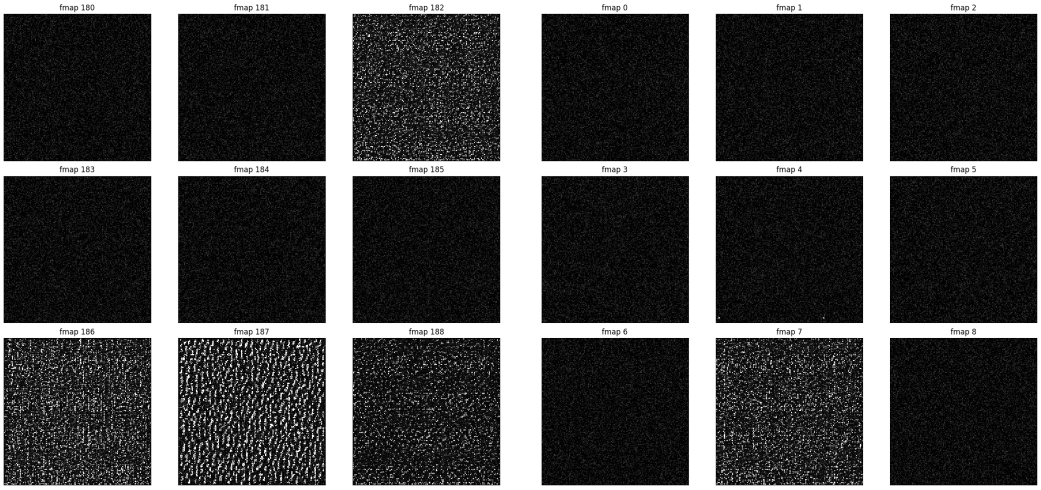


Figure 5.1: Visualization of features in PoseNet on small NORB- 'Before Inception'

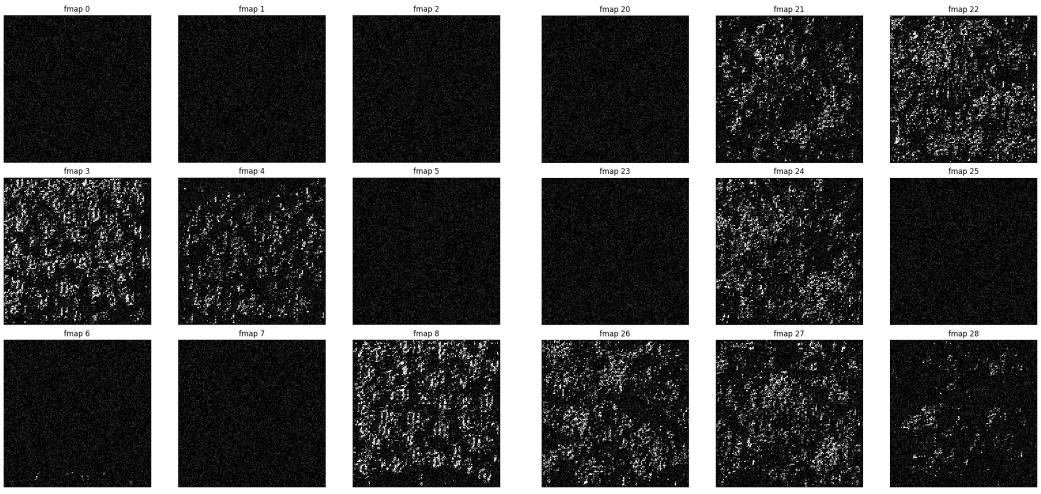


Figure 5.2: Visualization of features in PoseNet on small NORB
Left: Inception 3b, Branch 1x Right: Inception 5b, Branch 3x

Feature visualizations produced by convolutional filters in 'before inception' part of the network were not geometrically intuitive and seemed to detect textures at various scales. Further layers seemed to focus on specific patches in an image, rather than looking for definite lines or shapes.

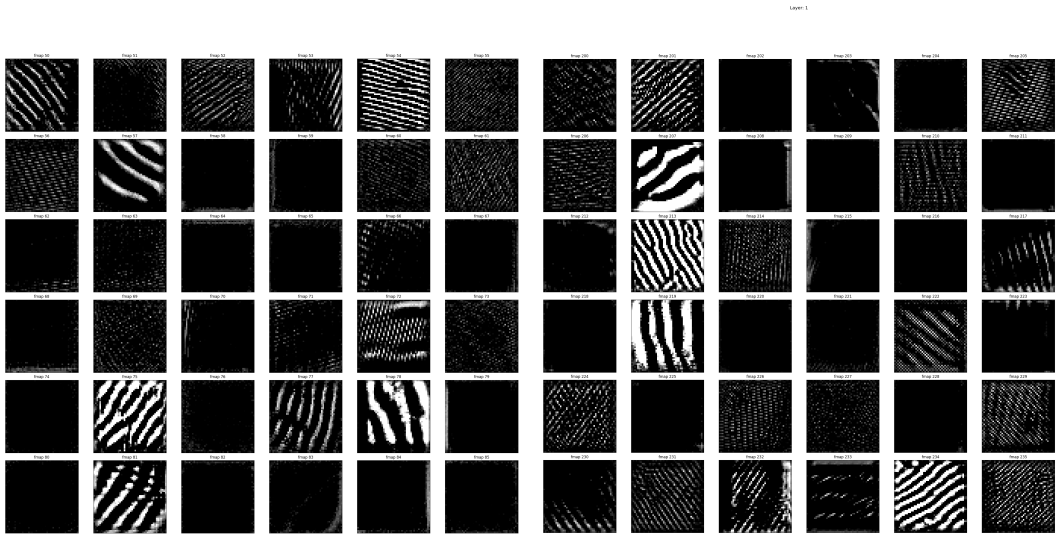


Figure 5.3: Visualization of features in PoseCap on small NORB- Conv layer
Left: filters 50-86 Right: filters 200-236

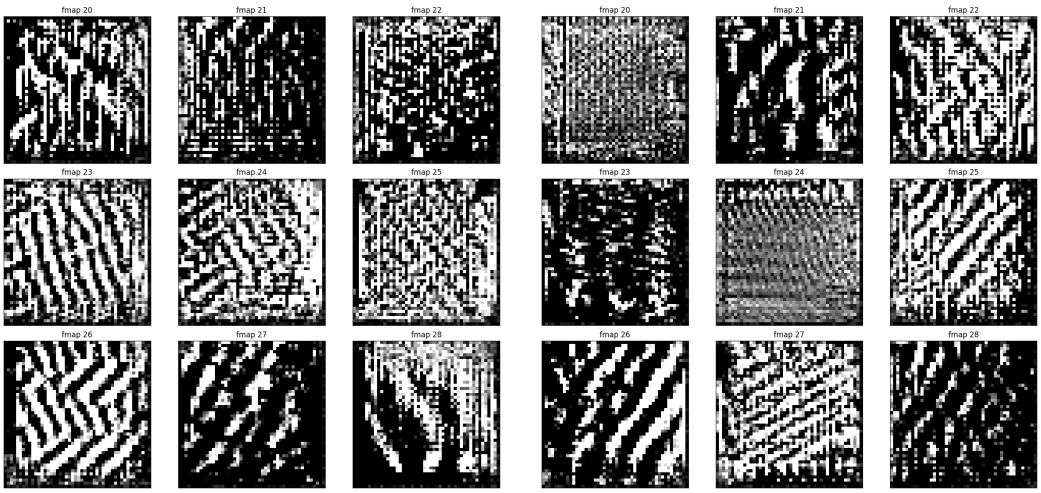


Figure 5.4: Visualization of features in PoseCap on small NORB- Primary Capsules:
Filters 20-56. Left: 1st Conv Right: 6th Conv

PoseCap successfully extracted lines of different thicknesses and directions as preliminary features for the network. Feature visualizations of primary capsules increased in complexity over the visualizations of previous layer, and also tried to detect certain patterns in an image. These patterns could have been formed from a combination of features detected in the previous layers. PoseCap produced more intuitive and geometrically relevant feature visualizations, which can be stated as the reason for its better performance.

Shop Facade

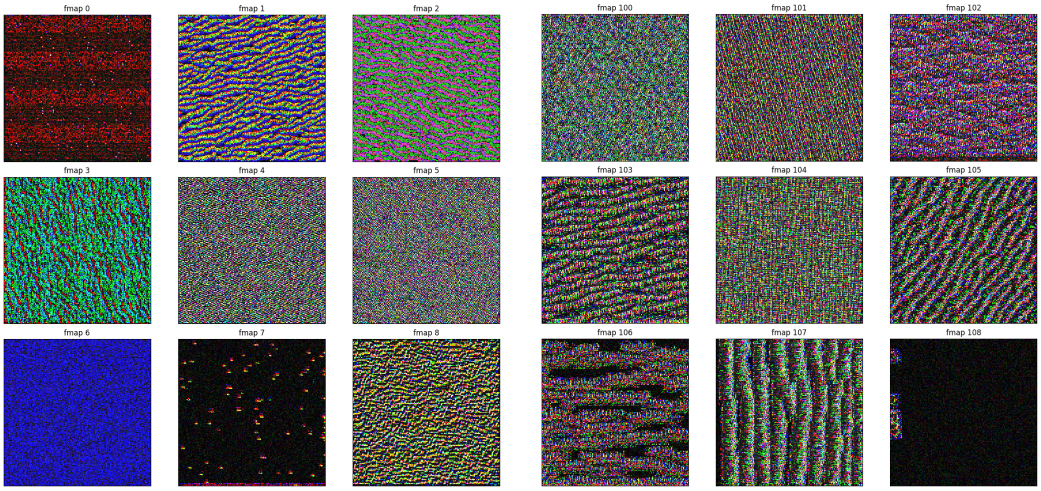


Figure 5.5: Visualization of features in PoseNet on Shop Facade- 'Before Inception'

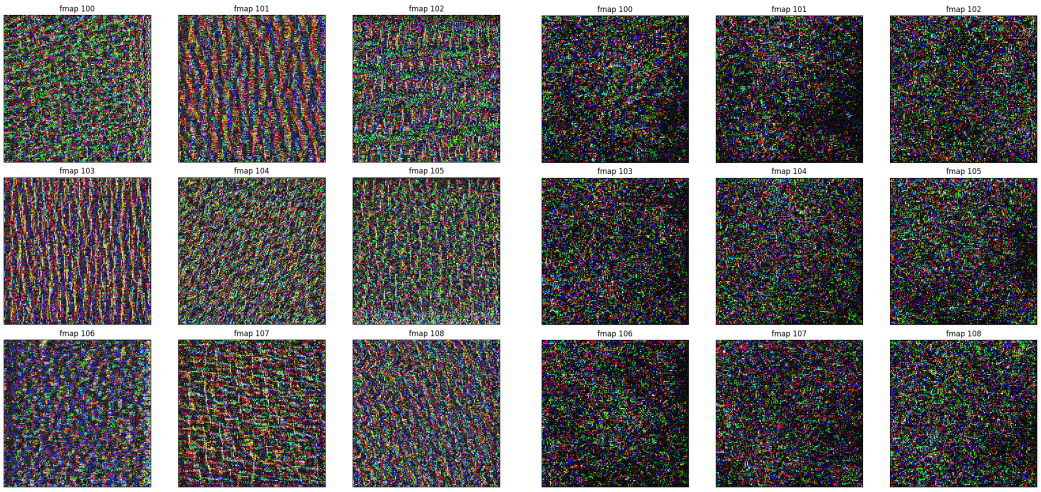


Figure 5.6: Visualization of features in PoseNet on Shop Facade
Left: Inception 3a, Branch 3x Right: Inception 5b, Branch 5x

Feature visualizations produced by PoseNet are picturesque. Each filter in the convolutional layers in 'before inception' part of the network is dominated by a distinctive colour and a specific pattern. The image turns kaleidoscopic and the patterns become complex as we go up the layers.

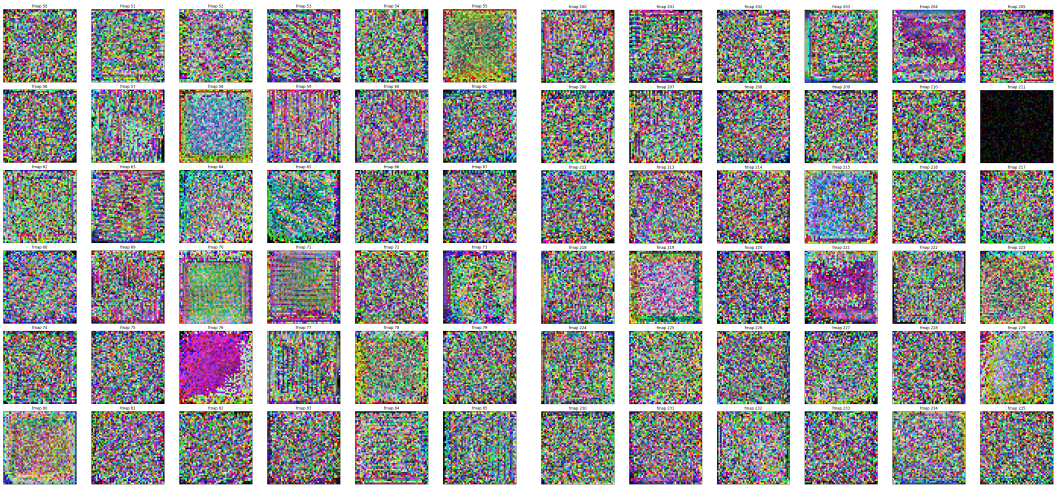


Figure 5.7: Visualization of features in PoseCap on Shop Facade- Conv layer: Left: filters 50-86 Right: filters 200-236

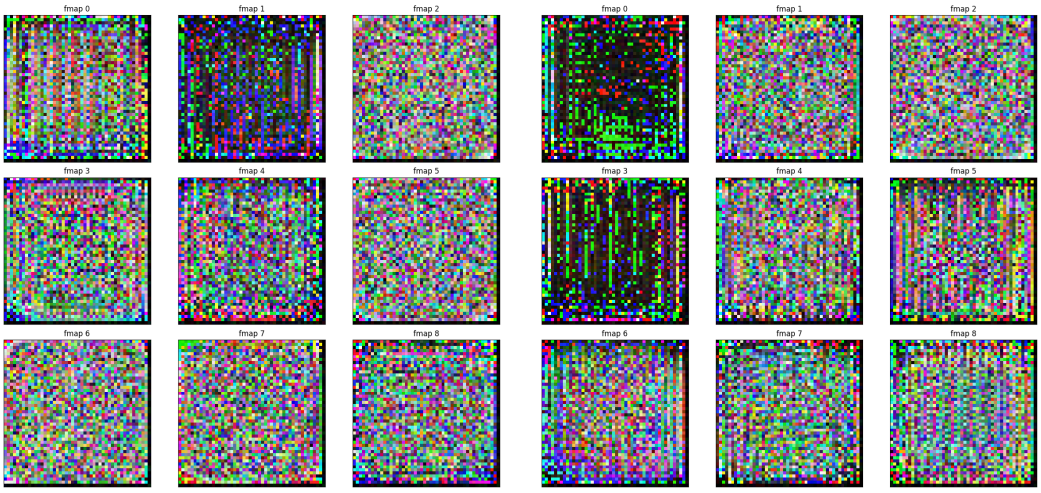


Figure 5.8: Visualization of features in PoseCap on Shop Facade- Primary Capsules: Filters 20-56. Left: 1st Conv Right: 6th Conv

In contrast, the filters in the convolutional layer of PoseCap did not seem to be dominated by a specific colour, but rather by a mix of colours, with concentrations in patches. Most filter visualizations also showed distinctive patterns. As we visualized activations in the primary capsule layer of the network, we observed an increase in the complexity of patterns, with no definite consistency in colour of the image.

In view of the feature visualizations obtained above, the following conclusions can be drawn.

- CNN-based networks, specifically PoseNet are over-dependent on colour and textures to infer pose of the camera, rather than the shape and size of objects in the images. This can be inferred from dismal performance of the network on small NORB, which consists of grey images, and better performance on multi-coloured and textured Shop Facade.
- PoseCap tried to extract edges and patterns from the image to infer pose of the camera, however did not use the colour and texture of the image effectively.
- Patterns in feature visualization for PoseNet were replicated throughout the image, indicating translational invariance, which was not the case with PoseCap.

In the above images for PoseCap, we visualized feature activations corresponding to layers in the primary capsule layer. But the only functional entity as per the model is a 7D Capsule, with 8192 such capsules in the primary capsule layer. Consequently, we considered what image will maximally activate 7 neurons corresponding to a capsule in the network, corresponding to both the datasets.

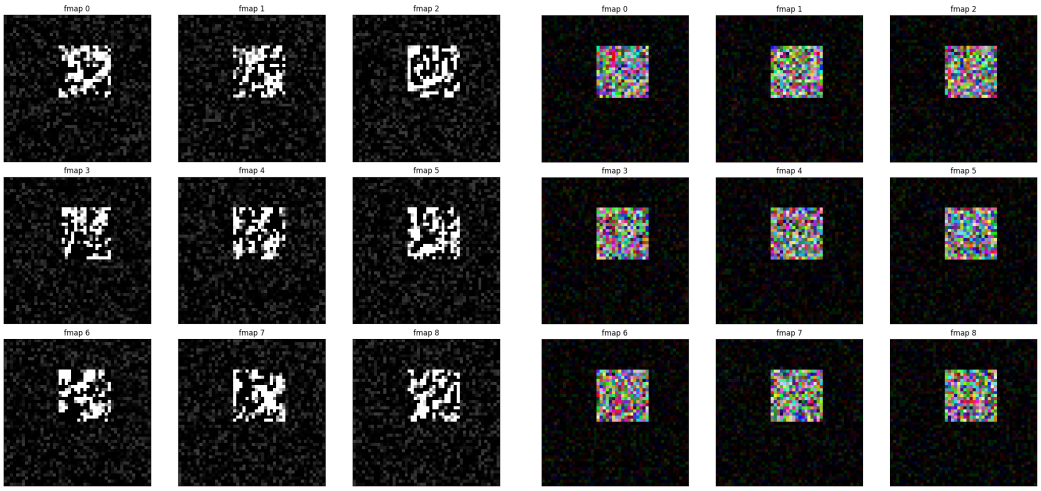


Figure 5.9: Visualization of features generated by a Capsule in the Primary Capsule layer. Left: Small NORB Right: Shop Facade

The local field of view of a capsule, in the primary capsule layer is visible in Fig.5.9. The activations of these capsules are generated through modified convolutions- where each convolution returns a 7D output corresponding to a capsule. This explained why the activations were localized to specific regions of the image.

5.2 Future Work

5.2.1 Understanding the Performance Variation in SFM and CNN Models with Datasets and Images

To understand the performance variation of SFM and CNN models, we can analyze the features and perform sensitivity analysis on a network which classifies images, to the model yielding a better result. For this purpose, we need to create a one-hot vector for every image, with one for the model which returns the best accuracy for it. The classification network which will again be a CNN can be ResNet18 by He *et al.* (2016) or even a network smaller than that.

Being open source, Active Search by Sattler *et al.* (2012) can be chosen as a representative for structure from motion models, and PoseNet for CNN-based models. The loss function could be chosen as binary cross-entropy loss between the model output and the one-hot target vector for every image.

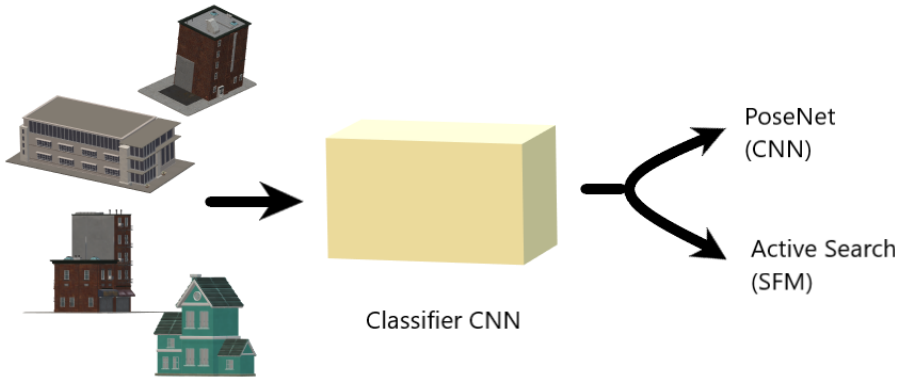


Figure 5.10: Network which classifies images to the model which yields a more accurate output

5.2.2 Capsule-based Networks- DR and EMCapsNet

We have experimented with our capsule based network- PoseCap, on small NORB and found better results than those obtained by PoseNet. However, the network performed inferior to PoseNet on Shop Facade. One of the reason could be the relatively small size of PoseCap. Hence a future line of work could be to expand the size of network

to test whether it performs better on more sophisticated and real life datasets like Shop Facade.

Additional improvements like incorporating geometrical constraints through the loss function such as reprojection error, could be made. Reprojection error takes into account the distance between the image projections of a 3D point made by the actual camera pose and the predicted one. This naturally weighs the position and orientation errors to suit the geometry of the scene. Kendall and Cipolla (2017) reported that L_1 norm for the loss function gave the best results.

$$\textit{Loss(I)} = \frac{\left\| \left(\sum_{g_i \in G} \pi(x, q, g_i) - \pi(\hat{x}, \hat{q}, g_i) \right) \right\|}{|G|}$$

where π is a function that maps a 3D point g to 2D image coordinates $(u, v)^{\mathrm{T}}$, and x and q are the position and orientation of the camera, respectively.

Retrieving g to calculate the projection using predicted camera pose can be done by the following formulation-

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = K(Rg + x)$$

where K (can be taken as Identity) is the camera calibration matrix, R is the camera rotation and

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{u'}{w'} \\ \frac{v'}{w'} \end{bmatrix}$$

5.2.3 Proposing a new model

In DRCapsNet, each capsule is a vector of instantiation parameters, whose length signifies its activation; a capsule in EMCapsNet on the other hand is a 4x4 matrix, along with a separate unit for activation. The weights which are used to calculate votes, from the capsule outputs, are modelled as a 4x4 matrix, encoding the transformation of instanti-

ation parameters. So a more intuitive model based on EMCapsNet would be to output the 4x4 camera pose transformation matrix for an image relative to another, rather than the camera pose straightaway.

The network will have similar branches for both the input images, and will output transformation matrix for the camera pose of the second image to the camera pose of the first image. The camera pose of the first image can then be calculated using the camera pose of the second image and the pose transformation matrix obtained. Only pairs of images which have points in common should be fed as the input to the network. For each image within the dataset, we can find a set of images which have points common to it. For a chosen image, we feed it into input one of the network and sequentially all the matching images from the set, into input two. Then the camera pose for the chosen image can be estimated as the average of the poseses estimated using the transformation matrix outputs of the network and the camera poseses of the image input two.

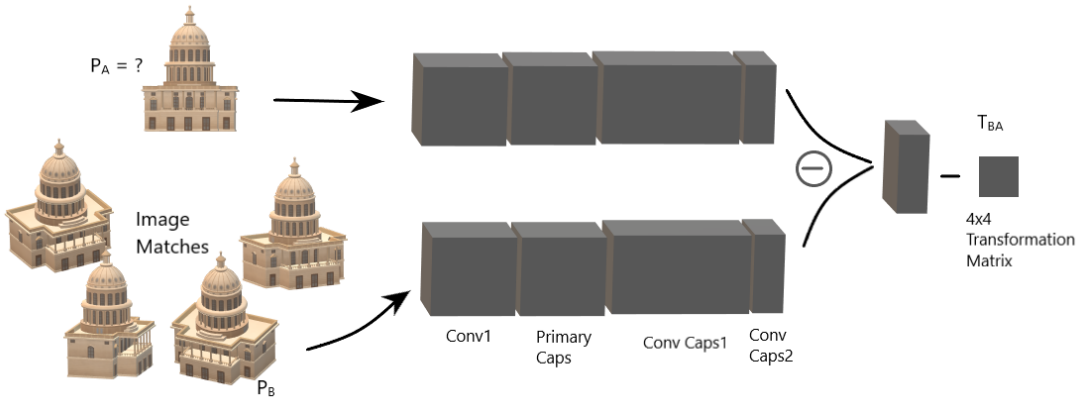


Figure 5.11: Proposed model using EMCapsNet and image matching.

As is a part of the Structure from Motion pipeline by Snavely *et al.* (2006), we can perform keypoint descriptor matching between all pairs of images using the approximate nearest neighbour library by Arya *et al.* (1998) and then robustly estimate a fundamental matrix for the pair using RANSAC (Fischler and Bolles (1981)). During each RANSAC iteration, a candidate fundamental matrix is computed using the eight-point algorithm (Hartley and Zisserman (2003)), followed by non-linear refinement. Finally, matches that are outliers to the recovered fundamental matrix are removed. This way, we can find images within the dataset with matching keypoint descriptors for each image.

5.3 Conclusion

This project concludes that Capsule based networks present a bright opportunity to overcome the deficiencies posed by CNNs, although further research is needed in this direction. Capsule-based networks lend themselves to better geometrical interpretation, and also detect lines and subsequently shapes and patterns in images, as exemplified by the feature visualizations. Camera pose prediction is the ideal platform to test the core hypothesis of capsules- which is that they extract part instantiation parameters from image pixel intensities and form 'wholes' from 'parts' based on the likenesses of parts to each other. We thus establish that Capsules perform well on the problem of camera pose estimation and can produce competitive results in future.

***

APPENDIX A

PoseCap based on EM Routing

A.1 EM Routing

We also modified our model to support routing by expectation-maximization. Using EM for routing activations from one layer to next, rather than dynamic routing offers numerous benefits like removing inefficiencies, and making the training easier, but, at the cost of making the model more complicated.

To effectively implement EMCapsNet for the task of pose prediction, we need to understand the EM routing process in detail, as it is susceptible to numerical instabilities. We discuss these instabilities after elaborating on the algorithm.

For CNNs, once trained, the network weights remain constant for each image, which implies that the network is static for each image. That is not the case for capsule-based networks. Here, although the weights remain the same once the network is trained, the assignment of 'capsules' in one layer to the next changes for each image.

Each image activates different capsules depending upon the features in it, which are then clustered together based upon their likeness, to form activations and instantiation parameters of the capsules in the next layer. We get an idea about the flow of the algorithm by looking at how the routing works between chosen two capsule layers *L* and *L+1*, for a single input image *I*.

We have the activations a_L^i and pose matrices M_L^i from layer *L*, and want to find the activations a_{L+1}^j and pose matrices M_{L+1}^j for layer *L+1*.

Each output capsule *j* is modelled as a 16 dimensional Gaussian μ_{L+1}^h and σ_{L+1}^h, where $1 < h < 16$. Each input pose matrix M_L^i is treated as a data point to be assigned to the Gaussians in the next layer. This presents an expectation-maximization problem- where we find μ_{L+1}^h, σ_{L+1}^h and a_{L+1} for all capsules *j* in the m step, using a_L, R_{ij} and V_L.

Here, the vote V_L is the product of the capsule pose matrix M_L^i and the weight *W*:

$V_{ij} = M_i W_{ij}$. R_{ij} is the posterior probability of a capsule *j*, given a vote V_i from capsule *i*. It is initialized to $\frac{1}{\omega_{L+1}}$.

The formulation for a_{L+1} is as follows:

$$a_j = sigmoid(\lambda(\beta_a - \sum_h cost_j^h))$$

where $cost_j^h$ is-

$$cost_j^h = \sum_i r_{ij} cost_{ij}^h \tag{A.1}$$

$$= \sum_i -r_{ij} \ln(p_{i|j}^h) \tag{A.2}$$

$$= \sum_i r_{ij} \big(\frac{(v_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2} + \ln(\sigma_j^h) + \frac{\ln(2\pi)}{2} \big) \tag{A.3}$$

$$= \frac{\sum_i r_{ij} (\sigma_j^h)^2}{2(\sigma_j^h)^2} + (\ln(\sigma_j^h) + \frac{\ln(2\pi)}{2}) \sum_i r_{ij} \tag{A.4}$$

$$= \big(\ln(\sigma_j^h) + \beta_v \big) \sum_i r_{ij} \quad \text{which } \beta_v \text{ is a constant} \tag{A.5}$$

β_v and β_a are trained discriminatively. λis an inverse temperature parameter increased by 1 after each routing iteration in our code implementation.

In the e-step of the algorithm, we calculate R_{ij}, using μ_{L+1}^h, σ_{L+1}^h and a_{L+1} calculated in the m-step. After a few iterations of the m-step and e-step, we have the resulting μ_{L+1}^h and σ_{L+1}^h. The 4x4 pose matrices M_{L+1}^i of layer *L+1* are formed from the 16 μ_{L+1}^h.

A.2 Model Architecture

ReLU Conv1 is a regular convolution (CNN) layer using a 5x5 filter with stride 2 outputting 32 channels using the ReLU activation. In PrimaryCaps, a 1x1 convolution filter is used to transform each of the 32 channels into 32 primary capsules. Each capsule contains a 4x4 pose matrix and an activation value. PrimaryCaps is followed by a convolution capsule layer ConvCaps1 using a 3x3 filters with stride 2. ConvCaps1 takes capsules as input and output capsules. ConvCaps1 is similar to a regular convolution layer except it uses EM routing to compute the capsule output. The capsule output of ConvCaps1 is then fed into ConvCaps2. ConvCaps2 is another convolution capsule layer but with stride 1. Finally, we have another convolution capsule layer, but with 1x1 kernel to output 1 capsule.

We then regress the final 16D pose matrix to a 2D output for elevation and azimuth. But as discussed in the model hypothesis of PoseCap model, the camera pose will be a function of the 'object' instantiation parameters. We had to multiply the outputs from DR-based PoseCap with some constants as the model could not learn values of less than one. We successfully overcame that problem for EM-based PoseCap. Similar to DR-PoseCap, we hypothesize that EM-PoseCap learns instantiation parameters for larger patches as we go deeper into the network.

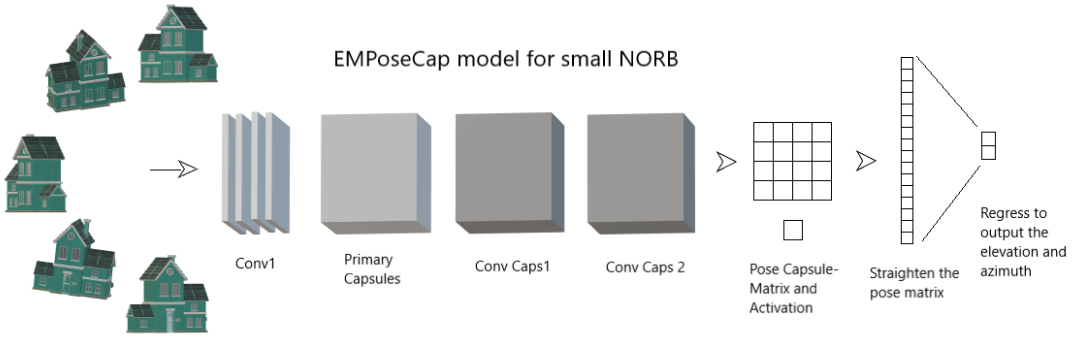


Figure A.1: EMPoseCap Architecture for Small NORB

Table A.1: Model Architecture Details

$LayerName$	$Details$
ReLU Conv1	Regular Convolution (CNN) layer using 5x5 kernels with 32 output channels, stride 2 and padding
PrimaryCaps	Modified convolution layer with 1x1 kernels, strides 1 with padding and outputing 32 capsules. Requiring 32x32x(4x4+1) parameters
ConvCaps1	Capsule convolution with 3x3 kernels, strides 2 and no padding. Requiring 3x3x32x32x4x4 parameters
ConvCaps2	Capsule convolution with 3x3 kernels, strides 1 and no padding
Pose Capsule	Capsule with 1x1 kernel. Requiring 32x1x4x4 parameters

A.3 Experiments and Result

Most open-source implementations suffer from several numerical instabilities. Gritzman (2019) mentions 3 such instabilities, such as when calculating the activation cost, as $\log(\sigma_j^h)$ is undefined at $\sigma_j^h = 0$. This happens when the parent capsule comprises only one child capsule, as the variance of the parent capsule is 0.

In the wake of COVID pandemic, we could not perform extensive testing to find the optimal hyperparameters. We have used a schedule of learning rates to train the model. We used 3 iterations of EM routing, although many implementations for smallNORB object detection achieve the best result using 2 iterations.

Comparison between EMPoseCap, DRPoseCap and PoseNet

DRPoseCap performed the best on small NORB, followed by EMPoseCap, and finally PoseNet. A possible reason for EMPoseCap's inferior performance to DRPoseCap could be sub-optimal hyperparameters. We have extensively tested DRPoseCap to find the optimal hyperparameters, while the same remains to be done for EMPoseCap. This suggests that capsule networks have the potential to out-perform CNNs in vision tasks.

Table A.2: Hyperparameters and Results for EMPoseCap

Hyperparameter	*Value*
Batch Size	3
Load Size	48
Fine Size	48
EM Iterations	3
lr1	0.001 - 250 Epochs
lr2	0.0001 - 100 Epochs
lr3	0.00005 - 100 Epochs
Epoch	445
Min Error	**16.26**

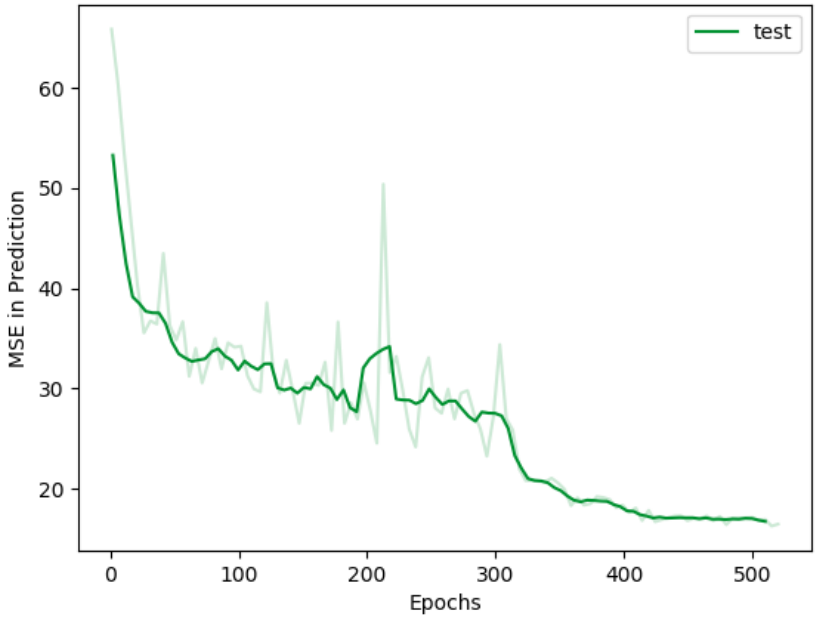


Figure A.2: Testing curve for EMPoseCap on Small NORB

Table A.3: Results on small NORB

PoseNet	*DRCapsNet*	*EMCapsNet*
44.67	**5.45**	16.26

REFERENCES

1. **Arya, S.**, **D. M. Mount**, **N. S. Netanyahu**, **R. Silverman**, and **A. Y. Wu** (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, **45**(6), 891–923. URL https://doi.org/10.1145/293347.293348.

2. **Brachmann, E.**, **A. Krull**, **S. Nowozin**, **J. Shotton**, **F. Michel**, **S. Gumhold**, and **C. Rother**, DSAC - differentiable RANSAC for camera localization. *In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017. URL https://doi.org/10.1109/CVPR.2017.267.

3. **Brahmbhatt, S.**, **J. Gu**, **K. Kim**, **J. Hays**, and **J. Kautz**, Geometry-aware learning of maps for camera localization. *In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Brahmbhatt_Geometry-Aware_Learning_of_CVPR_2018_paper.html.

4. **Cignoni, P.**, **M. Callieri**, **M. Corsini**, **M. Dellepiane**, **F. Ganovelli**, and **G. Ranzuglia**, MeshLab: an Open-Source Mesh Processing Tool. *In* **V. Scarano**, **R. D. Chiara**, and **U. Erra** (eds.), *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. ISBN 978-3-905673-68-5.

5. **Cohen, T.** and **M. Welling**, Group equivariant convolutional networks. *In* **M. Balcan** and **K. Q. Weinberger** (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/cohenc16.html.

6. **Donoser, M.** and **D. Schmalstieg**, Discriminative feature-to-point matching in image-based localization. *In 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE Computer Society, 2014. URL https://doi.org/10.1109/CVPR.2014.73.

7. **Fischler, M. A.** and **R. C. Bolles** (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, **24**(6), 381–395. URL http://doi.acm.org/10.1145/358669.358692.

8. **Gritzman, A. D.**, Avoiding implementation pitfalls of "matrix capsules with EM routing" by hinton et al. *In* **A. Zeng**, **D. Pan**, **T. Hao**, **D. Zhang**, **Y. Shi**, and **X. Song** (eds.), *Human Brain and Artificial Intelligence - First International Workshop, HBAI 2019, Held in Conjunction with IJCAI 2019, Macao, China, August 12, 2019, Revised Selected Papers*, volume 1072 of *Communications in Computer and Information Science*. Springer, 2019. URL https://doi.org/10.1007/978-981-15-1398-5_16.

9. **Hartley, R.** and **A. Zisserman**, *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2003, 2 edition. ISBN 0521540518.

10. **He, K.**, **X. Zhang**, **S. Ren**, and **J. Sun**, Deep residual learning for image recognition. *In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016. URL https://doi.org/10.1109/CVPR.2016.90.

11. **Hinton, G. E.**, **A. Krizhevsky**, and **S. D. Wang**, Transforming auto-encoders. *In* **T. Honkela**, **W. Duch**, **M. A. Girolami**, and **S. Kaski** (eds.), *Artificial Neural Networks and Machine Learning - ICANN 2011 - 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I*, volume 6791 of *Lecture Notes in Computer Science*. Springer, 2011. URL https://doi.org/10.1007/978-3-642-21735-7_6.

12. **Hinton, G. E.**, **S. Sabour**, and **N. Frosst**, Matrix capsules with EM routing. *In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=HJWLfGWRb.

13. **Hui, J.** (2017). Understanding matrix capsules with em routing (based on hinton's capsule networks). URL https://jhui.github.io/2017/11/14/Matrix-Capsules-with-EM-routing-Capsule-Network/.

14. **Jaderberg, M.**, **K. Simonyan**, **A. Zisserman**, and **K. Kavukcuoglu**, Spatial transformer networks. *In* **C. Cortes**, **N. D. Lawrence**, **D. D. Lee**, **M. Sugiyama**, and **R. Garnett** (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 2015. URL http://papers.nips.cc/paper/5854-spatial-transformer-networks.

15. **Jégou, H.**, **M. Douze**, **C. Schmid**, and **P. Pérez**, Aggregating local descriptors into a compact image representation. *In The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*. IEEE Computer Society, 2010. URL https://doi.org/10.1109/CVPR.2010.5540039.

16. **Kendall, A.** and **R. Cipolla** (2017). Geometric loss functions for camera pose regression with deep learning. *CoRR*, **abs/1704.00390**. URL http://arxiv.org/abs/1704.00390.

17. **Kendall, A.**, **M. Grimes**, and **R. Cipolla**, Posenet: A convolutional network for real-time 6-dof camera relocalization. *In 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015. URL https://doi.org/10.1109/ICCV.2015.336.

18. **LeCun, Y.**, **F. J. Huang**, and **L. Bottou**, Learning methods for generic object recognition with invariance to pose and lighting. *In 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), with CD-ROM, 27 June - 2 July 2004, Washington, DC, USA*. IEEE Computer Society, 2004. URL http://doi.ieeecomputersociety.org/10.1109/CVPR.2004.144.

19. **Melekhov, I.**, **J. Ylioinas**, **J. Kannala**, and **E. Rahtu**, Image-based localization using hourglass networks. *In 2017 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017. URL https://doi.org/10.1109/ICCVW.2017.107.

20. **Milford, M.** and **G. F. Wyeth**, Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. *In IEEE International Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*. IEEE, 2012. URL https://doi.org/10.1109/ICRA.2012.6224623.

21. **Olah, C.**, **A. Mordvintsev**, and **L. Schubert** (2017). Feature visualization. *Distill*. Https://distill.pub/2017/feature-visualization.

22. **Sabour, S.**, **N. Frosst**, and **G. E. Hinton**, Dynamic routing between capsules. *In* **I. Guyon**, **U. von Luxburg**, **S. Bengio**, **H. M. Wallach**, **R. Fergus**, **S. V. N. Vishwanathan**, and **R. Garnett** (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017. URL http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.

23. **Sattler, T.**, **B. Leibe**, and **L. Kobbelt**, Improving image-based localization by active correspondence search. *In* **A. W. Fitzgibbon**, **S. Lazebnik**, **P. Perona**, **Y. Sato**, and **C. Schmid** (eds.), *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I*, volume 7572 of *Lecture Notes in Computer Science*. Springer, 2012. URL https://doi.org/10.1007/978-3-642-33718-5_54.

24. **Sattler, T.**, **Q. Zhou**, **M. Pollefeys**, and **L. Leal-Taixé**, Understanding the limitations of cnn-based absolute camera pose regression. *In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Sattler_Understanding_the_Limitations_of_CNN-Based_Absolute_Camera_Pose_Regression_CVPR_2019_paper.html.

25. **Shavit, Y.** and **R. Ferens** (2019). Introduction to camera pose estimation with deep learning. *CoRR*, **abs/1907.05272**. URL http://arxiv.org/abs/1907.05272.

26. **Sivic, J.** and **A. Zisserman**, Video google: A text retrieval approach to object matching in videos. *In 9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*. IEEE Computer Society, 2003. URL https://doi.org/10.1109/ICCV.2003.1238663.

27. **Snavely, N.**, **S. M. Seitz**, and **R. Szeliski** (2006). Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, **25**(3), 835–846. URL https://doi.org/10.1145/1141911.1141964.

28. **Szegedy, C.**, **W. Liu**, **Y. Jia**, **P. Sermanet**, **S. E. Reed**, **D. Anguelov**, **D. Erhan**, **V. Vanhoucke**, and **A. Rabinovich**, Going deeper with convolutions. *In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015. URL https://doi.org/10.1109/CVPR.2015.7298594.

29. **Torii, A.**, **R. Arandjelovic**, **J. Sivic**, **M. Okutomi**, and **T. Pajdla**, 24/7 place recognition by view synthesis. *In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015. URL https://doi.org/10.1109/CVPR.2015.7298790.

30. **Walch, F.**, **C. Hazirbas**, **L. Leal-Taixé**, **T. Sattler**, **S. Hilsenbeck**, and **D. Cremers** (2016). Image-based localization with spatial lstms. *CoRR*, **abs/1611.07890**. URL http://arxiv.org/abs/1611.07890.