

Satellite Pose Estimation using Convolutional Neural Networks

Shehzeen Hussain*, Chinmayee Bhanu*, Arunkumar Ravichandran* and Sneha Kondur*

*Department of Electrical and Computer Engineering

University of California, San Diego

ECE 228 Group 23

Project Github link

ssh028@ucsd.edu, cbhanu@ucsd.edu, arravich@ucsd.edu, skondur@ucsd.edu

Abstract—Pose estimation of known uncooperative spacecraft plays an important role in various satellite servicing missions of scientific, economic, and societal benefits. In this work, we aim to estimate the pose, i.e., the relative position and attitude, of a known spacecraft from individual grayscale images using deep neural networks. Convolutional neural networks have been widely exploited for image processing tasks such as object detection, classification, human pose estimation and action recognition. Our results on various methods incorporating these CNNs are in the top-12 of the Kaggle competition from which the dataset is obtained with a least error for synthetic image of 1.291.

I. INTRODUCTION

Satellite pose estimation is essential for uncooperative spacecraft servicing missions and automated removal of debris. Today these services are organized by various providers such as RemoveDEBRIS by Surrey Space Centre, Restore-L by NASA, Phoenix program by DARPA, and several missions planned by new space start-ups such as Effective Space Solutions and Infinite Orbits. They utilize satellite pose estimation to ensure safety of satellites and also refurbishment of expensive space assets. Additionally, accurate pose estimation would enable autonomous docking which can advance current travel missions to aim for more distant destinations. In this project, we aim to estimate the pose, i.e., the relative position and attitude, of a known spacecraft from individual grayscale images. Prior demonstrations of pose estimation have utilized image processing based on hand-engineered features and/or a-priori coarse knowledge of the pose [1]–[3]. However, these approaches are not scalable to spacecraft of different physical properties as well as not robust to the dynamic illumination conditions of space. Moreover, a-priori knowledge of the pose is not always available nor desirable when full autonomy is required.

In this work, we study the use of deep Convolutional Neural Networks (CNNs) [4], [5] for pose determination in spaceborne applications. Recent advances in machine learning have demonstrated CNNs to be effective for various image processing tasks [6]–[12] on unseen datasets. CNNs are able to automatically extract relevant features from the images for a given task (such as object recognition) that in traditional algorithms had to be hand-engineered. CNNs take advantage of local spatial coherence in images, which allow them to have fewer weights as some parameters are shared. This process,

taking the form of convolutions, makes them especially well suited to extract relevant information at a low computational cost. CNNs have the advantage of being scale and spatially invariant and can be trained using augmented data to account for varying conditions. Hence, a CNN based pose determination technique is scalable to spacecraft of different structural and physical properties as well as robust to the dynamic illumination conditions of space. The input to our algorithm is an image of a satellite. We then use a deep CNN to output a predicted pose, which is a 7-dimensional vector consisting of position, a 3D vector of x, y, z coordinates and orientation via a 4D unit Quaternion representation).

II. BACKGROUND

In this section, we provide a background on the motion and pose of rigid bodies such as satellites in space. In general, the motion of a mobile object is completely specified by the motion of a local reference frame called the body frame. A body frame is described by the position of its origin of coordinates, and its orientation relative to another frame called the world frame. The world frame of reference is usually considered static and attached arbitrarily somewhere in the world space. A combination of position and orientation is referred to as the pose of an object. The pose of a satellite S relative to the world frame of reference F is denoted as S_F and is specified by its position and orientation $S_F = (P, \phi)$, where P is a translation vector indicating the position of the origin of S in frame F and ϕ is an orientation of frame S with respect to frame F . In this work, the position P of the satellite is represented using a three dimensional vector of Cartesian coordinates (x, y, z) . The orientation is represented using a unit quaternion representation $\phi = (\phi_w, \phi_x, \phi_y, \phi_z)$ where, ϕ_w is a real scalar and ϕ_x, ϕ_y, ϕ_z is a 3D vector in complex space. Therefore, the pose of the satellite is described using a 7 dimensional vector $(x, y, z, \phi_w, \phi_x, \phi_y, \phi_z)$.

III. RELATED WORK

The contemporary state-of-the-art monocular pose estimation algorithms for spaceborne applications have utilized image processing techniques based on hand-engineered features and a-priori knowledge of the pose [1]–[3], [13]–[16]. The

hand-engineered features are then compared against a reference texture model of the satellite to determine its pose. Since the method of using hand engineered features is not robust to the dynamic illumination conditions of space and cannot be scaled to spacecrafts of different physical and structural properties, recent advancements in satellite pose estimation techniques have greatly relied on deep learning algorithms. Broadly, these algorithms bypass the classical image processing-based techniques and instead attempt to learn the non-linear transformation between the two-dimensional input image space and the output pose space in an end-to-end fashion.

The deep learning based methods either discretize the pose space and solve the resulting classification problem [17], [18] or directly regress the relative pose from the input image [19]–[21]. Classification-based approaches rely on the fine discretization of the pose space into a large number of pose labels in order to achieve reasonable pose accuracy. On the other hand, direct regression-based approaches require careful choice of parameters to avoid unpredictable behavior while learning the transformation between the input two-dimensional pixel information and the output regression parameters describing the seven-dimensional pose space.

Deep Convolutional Neural Network (CNN) based pose determination method is implemented in [22]. It proposes a method to discretize the pose space and train the CNN with train set images and their respective pose labels. Their approach consists of an offline training phase and an online prediction phase. During the training phase, the method automatically generates several thousand synthetic images of a target spacecraft and uses them to train a CNN. During the prediction phase, the trained CNN is used to predict a pose label of a target satellite image that is taken at close proximity.

Designing deep neural networks is particularly difficult because the training error increases with the increase in depth of the network. A solution to avoid this problem is to add layers that can be constructed as identity mappings. The resulting deeper model shall have training error no greater than its shallower counterpart. [23] introduces a deep residual learning framework called the ResNet architecture that helps to ease the training of deep networks for the task of image classification. Instead of letting every few stacked layers directly fit a desired underlying mapping $H(x)$, the authors explicitly allow these layers to fit a residual mapping $F(x) = H(x) - x$ as shown in Fig 2. The ResNet architecture has different variants such as ResNet18, ResNet34, ResNet50, ResNet101 and ResNet152 depending on the number of layers. With the increase in the number of layers, the network can extract more informative features. The authors of [23] demonstrate that extreme deep residual nets are easy to optimize when compared to the counterpart of a simple stacked network that exhibits higher training error when depth increases.

IV. DATASET

We utilize the dataset provided by Space Rendezvous Laboratory (SLAB) for their Satellite Pose Estimation Challenge

on Kaggle. The training dataset consists of 12000 synthetic images of a satellite with corresponding ground truth pose labels. The test dataset consists of 2998 synthetic images and 300 real images. The real images, which vary significantly from the synthetic ones, are collected with a Tango satellite mockup at SLAB, and is essentially provided to evaluate the transferability of the pose estimation model and algorithm on a realistic dataset.

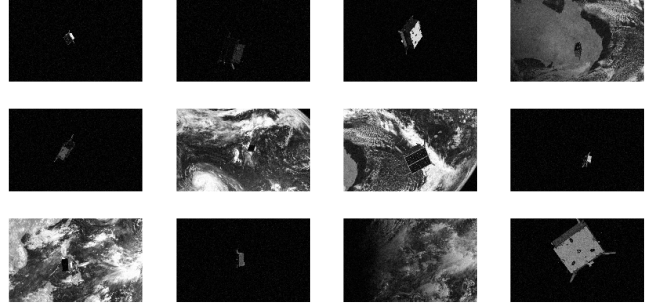


Fig. 1: Satellite image dataset provided by SLAB

All the images provided are 8 bit monochrome with a resolution of 1920x1080 pixels. The synthetic images are generated with SLAB’s Optical Simulator [24], that uses a high fidelity texture model of the Tango spacecraft of the PRISMA mission [15], [25] and the camera model of the Point Grey Grasshopper 3 camera with a Xenoplan 1.9/17mm lens (VBS). Gaussian blurring and white noise are added to all images to emulate the depth of field and shot noise, respectively. A portion of the images have a black background, simulating situations where the target is imaged against the star-field. The rest of the images have a background covered partially or fully by real images of the Earth. The real images, which form the second set of test images, come from TRON facility at SLAB. TRON provides images of a 1:1 mockup model of the Tango spacecraft of the PRISMA mission [25] using an actual Point Grey Grasshopper 3 camera with a Xenoplan 1.9/17mm lens. Note that this is the same camera as used in the OS camera emulator software. Calibrated motion capture cameras report the positions and attitudes of the camera and the Tango spacecraft, which are then used to calculate the ground truth pose of Tango satellite with respect to the camera. The test set of real images are used to evaluate the transferability of each of our models from synthetic to real images.

Since we utilize pre-trained models in our experiments, we use the same preprocessing as the pre-trained networks, which were trained on ImageNet dataset [26]. We resize all images using bilinear interpolation to size 224 x 224 and normalize all channels with the mean and standard deviation of ImageNet dataset.

V. METHODOLOGY

Our primary objective is to train a deep neural network on the synthetic image data set, that is capable of estimating the 7D pose of the satellite on unseen images in the test data set.

In this work, we begin by defining a baseline model with which we compare the results achieved by the different methods and models that we use. The baseline model is a part of an ongoing Kaggle competition.

A. Convolutional Neural Networks

1) **Convolutional Layer:** Convolutional layers form a set of small learnable spatial filters which extend through the full depth of the input volume. During the forward pass, we slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. Intuitively, the learned filters activate when they see visual features like an oriented edge or a coloured blotch in the initial layers or eventually entire honeycomb or wheel-like patterns on higher layers of the network.

$$h_{i,j} = \sum_{k=1}^m \sum_{l=1}^m w_{k,l} * x_{i+k-1,j+l-1}$$

where m - kernel width and height, h - convolution output, x - input, w - convolution kernel weights [27]

2) **Max-Pool Layer:** The function of pooling layer [27] is to reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. The max-pooling layer outputs the maximum value of the input, that falls within the kernel.

$$h_{i,j} = \max\{x_{i+k-1,j+l-1}\}, \forall 1 \leq k \leq m \text{ and } 1 \leq l \leq m$$

3) **Fully-connected Layer:** The fully connected layer [27] is a linear layer after all the convolutional layers. This layer is characterized by a weight matrix $W_{channels \times OutputSize}$ and a bias vector $b_{1 \times OutputSize}$. The fully connected layer performs the following operation on $ConvOut$: the output of the last convolution layer:

$$FinalOutput = ConvOut \times W + b$$

B. ResNet Architecture

It was seen that an increase in depth proved beneficial for any task employing deep neural nets. But, as the network depth increases, the accuracy got saturated and then degraded rapidly. Gradient information can get lost as we pass through multiple layers as we have problems of vanishing gradients. Resnets introduce an identity shortcut connection that skips one or more layers, as shown in the figure while keeping the degradation same as it's shallower counterpart. Skip connections also help traverse information faster in deep neural networks.

$$y = F(x, W_i) + x \quad (1)$$

where W_i are the weights of the weight layer and 'x' is the identity connection. We use different architectures using these residual connections and document the differences in them.

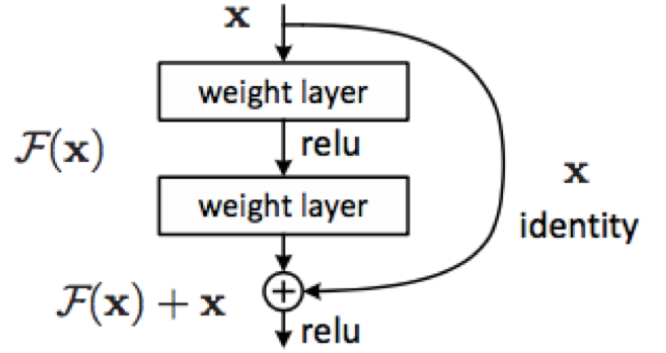


Fig. 2: Residual Learning Building Block

C. ResNet-18 for Satellite Pose Estimation

Transfer Learning allows us to use an existing model, trained on very large scale datasets, as feature extractors that can be used to learn the underlying model of the dataset of our new task. The new task entails estimating the pose and orientation of satellites in the SLAB dataset, using a pre-trained ResNet-18 [23] model. We use the ResNet-18 model pretrained on ImageNet [26], which contains 1.2 million images with 1000 categories, to estimate the position and orientation of satellite in our dataset. For localization and orientation regression to work with limited data we leverage the powerful representations learned from large classification datasets like the ImageNet, by pre-training the weights on these datasets.

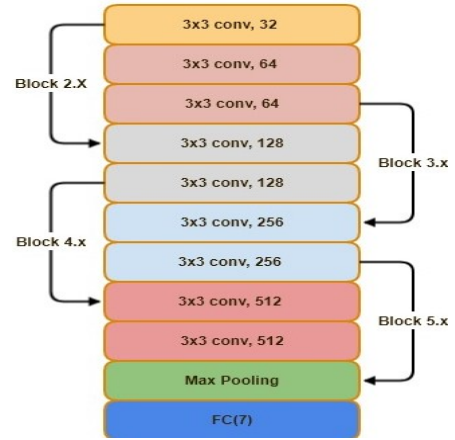


Fig. 3: ResNet-18 network for satellite pose estimation

We begin by analyzing the performance of the ResNet-18 model in estimating pose and orientation for the satellites of SLAB dataset in terms of accuracy over number of epochs. In order to do this, we create a new model by replacing the Softmax layer of the ResNet-18 with our own Fully-Connected layer to predict 7 dimensional vector of our orientation and position. This Fully-Connected layer serves as a regression layer which is designed to output a 7 dimensional pose vector representing position (3 dimensions) and orientation (4 dimensional quaternion). We freeze the weights of all the

layers and train only the last Fully-Connected layer of the new model on SLAB dataset to evaluate performance. By freezing layers, we do not change the weights during gradient descent/optimization of the specific layers.

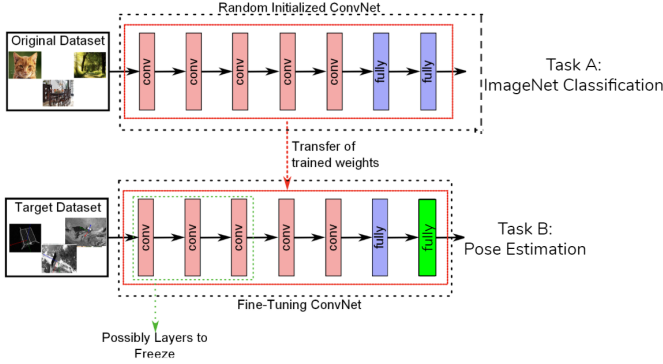


Fig. 4: Transfer learning using the weights from Imagenet dataset and freezing the lower layer weights and only updating the fully-connected regression layer

D. YOLO

We did transfer learning from weights trained on the ImageNet dataset for the task of image classification. We now try asking the question, will using an architecture or weights trained specifically to the task of localization help? We thus use weights adapted from the YOLOv3 [28] Darknet53 architecture, which was trained on the COCO dataset [29], after initializing it with ImageNet trained weights [26]. Since YOLO performs the work of correctly identifying the 2-D location points within the image, we incorporate these weights and modify the last layer to again regress to a 7-D vector instead of the bounding boxes of YOLO.

VI. EXPERIMENTS

We implement a variety of models to study the effect of various optimization techniques in isolation and in combination with other techniques. In this section we describe each of our experiments and study their effects on performance. For all our experiments, we use an optimizer to implement stochastic gradient descent with momentum and a learning rate of 0.001. We varied many hyper-parameters including the network structure and analyzed them. We used a batch size of 32 since it gave us sufficiently accurate results. We also present our evaluation metrics for measuring error for our regression task.

A. Evaluation Metrics

Evaluation of the total error on synthetic image test set and the real image test set gives the synthetic image error and real image error respectively. For the designed experiments, we report the total error on both the real and synthetic image Test datasets in Table I. The total error in estimating the 3D pose is calculated as the addition of the position error and the orientation error. [30] The position error for an image

Type	Filters	Size	Output
Convolutional	32	3 × 3	256 × 256
Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1
	Convolutional	64	3 × 3
Residual			128 × 128
Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1
	Convolutional	128	3 × 3
Residual			64 × 64
Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1
	Convolutional	256	3 × 3
Residual			32 × 32
Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1
	Convolutional	512	3 × 3
Residual			16 × 16
Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1
	Convolutional	1024	3 × 3
Residual			8 × 8
Avgpool			Global

Fig. 5: Darknet53 architecture [28] for feature extraction in the YOLOv3. We add fully connected layers to regress a 7D pose.

I is simply the L2-norm of the difference in estimated and ground truth position vectors, normalized with the ground truth distance of the satellite.

$$error_{position}^{(i)} = \frac{|P_{gt}^{(i)} - P_{est}^{(i)}|_2}{|P_{gt}^{(i)}|_2} \quad (2)$$

The orientation error is defined as the angle of rotation that aligns the estimated and ground truth orientations.

$$error_{orientation}^{(i)} = 2 \cdot \arccos(|\langle \phi_{est}^{(i)}, \phi_{gt}^{(i)} \rangle|) \quad (3)$$

The pose error for an image is the sum (1-norm) of the orientation and position errors.

$$error_{pose}^{(i)} = error_{position}^{(i)} + error_{orientation}^{(i)} \quad (4)$$

Let the number of images in the test set be N . The total error is the average of pose errors over all N images of the test set.

$$error = \frac{1}{N} \sum_{i=1}^N error_{pose}^{(i)} \quad (5)$$

B. Increasing the layers

We increased the number of layers in our Resnet architecture and observed improved results with 50 layers. **ResNet-50** outperforms ResNet-18 on the original ImageNet classification task [23] and we observed that it improves our results also. This is shown in Table I where Model B and C that have 50 layers, outperform Baseline Model, even though they all utilize MSE loss. Additionally, ResNet-50 is a deeper network (50 layers) than ResNet-18 (18 layers) so it extracts

more informative features. This is because having residual connections facilitates overcoming the disadvantages of deeper convolutional networks [23] where the model always learns residual functions with increasing depth; and all information is always passed through, with additional residual functions to be learned.

We have also utilized **Densenet-121** architecture [31] to estimate the satellite pose. Densenet-121 has 121 layers and carries forward inputs from all previous layers. The results of this architecture are depicted as Model D in Table I.

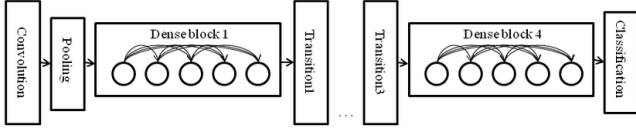


Fig. 6: Densenet network with inputs from previous layers [31]

C. Varying loss functions

For our baseline model, we utilize MSE as our loss function. By experimenting with different loss functions, we were able to improve our results.

1) **MSE Loss vs L1 Loss:** Target value is y_i and corresponding estimated value $h(x_i)$, where x_i denotes the feature set of a single sample out of n samples present. L1 Loss function minimizes the absolute differences between the estimated values and the existing target values.

$$S = \sum_{i=0}^n |y_i - h(x_i)| \quad (6)$$

whereas MSE loss function minimizes the squared differences between the estimated and existing target values [32].

$$S = \sum_{i=0}^n (y_i - h(x_i))^2 \quad (7)$$

L1 loss function is more robust and is generally not affected by outliers. MSE loss function will try to adjust the model according to these outlier values and it's highly sensitive to outliers in the dataset. Thus we used L1 loss function to increase the accuracy of our results.

2) **Posenet:** Taking inspiration from [20], we plan to use a joint learning architecture for branching out the estimation for position and orientation. This architecture is then tested by using two different losses, the generic MSE Loss and a PoseLoss, especially designed to balance between the expected value of position and orientation errors. The optimal scaling factors α and β were found out by doing a grid search over the validation split of the dataset. The prediction of the quaternion was also normalized to a unit quaternion.

$$loss(I) = (e^{-\alpha} + 1) \| \hat{x} - x \|_1 + (e^{-\beta} + 1) \| \hat{q} - \frac{q}{\|q\|} \|_1 \quad (8)$$

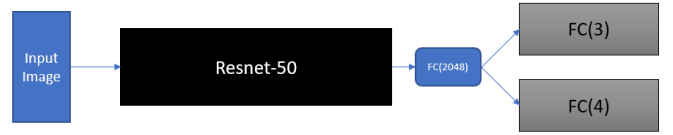


Fig. 7: Modified Resnet-50 architecture with 2 branched out fully-connected layers for estimating position and orientation separately

We initiate the shared base network with ResNet-50 weights and the branches with random initialization. We then train the entire network end to end on Euclidean loss using stochastic gradient descent with the following objective loss function. [20]

3) **Linearly Weighted Loss Function:** Utilizing another proposed method from Posenet paper [20], we combine position and orientation into a single loss function with a linear weighted sum as shown below:

$$L_{\beta(I)} = L_{x(I)} + \beta L_{q(I)} \quad (9)$$

Because x (position) and q (orientation) are expressed in different units, a scaling factor, β , is used to balance the losses. This hyperparameter is used to keep the expected value of position and orientation errors approximately equal. We perform significant tuning over the hyperparameter β to get the lowest average error in pose estimation over the unseen synthetic image dataset. This is demonstrated in Fig 8 where we find that the model with $\beta = 22$ gives the lowest error.

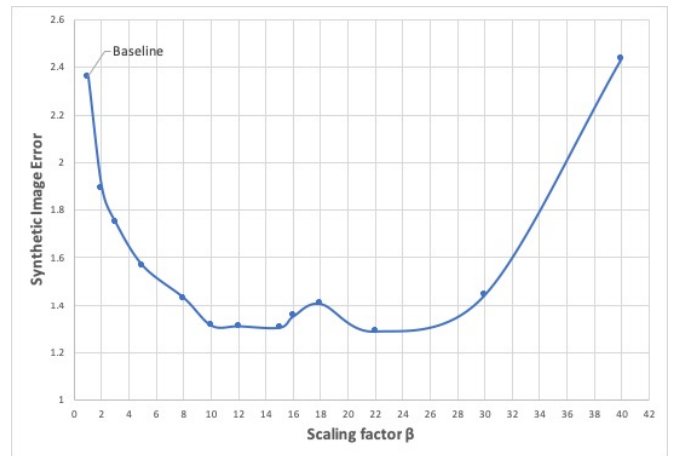


Fig. 8: Synthetic Image Error for a single model with a range of scale factors β . This experiment was conducted to find the optimum value for β which gives the lowest error in pose estimation over the unseen synthetic image dataset.

VII. RESULTS

In this section, we evaluate the effect of our experiments, namely different loss functions, freezing layer weights, different architectural changes and using randomly initialized networks. We have submitted all of our models to the Kaggle

Model Name	Description	Real Image Error	Synthetic Image Error
Baseline	ResNet-18 + MSE Loss	2.663	2.336
A	Baseline with Random Initialization	4.581	2.418
B	Freezing layers ResNet-50	2.569	2.14
C	Transfer Learning with ResNet-50	1.961	2.195
D	Densenet-121	2.45	2.24
E	L1 Loss + ResNet-50	2.218	2.082
F	Linearly Weighted Loss + ResNet-50	1.621	1.291
G	Branched Resnet-50 with PoseNet loss	1.859	1.722
H	DarkNet-53(YOLOv3) with PoseNet loss	2.61	2.14

TABLE I: Error values for Real images test dataset and synthetic images test dataset for different experiments carried out. Lower error values are better. All ResNet models are pre-trained on ImageNet except Model A. Our best result highlighted in bold.

competition in order to have a definitive comparison of the performances of different models. We report our results on Table I and reference this table throughout the section.

A. Comparison with Randomly Initialized Network

We find that using ResNet models pre-trained on ImageNet, yield higher accuracies than using randomly initialized ResNet models for pose estimation. This is possibly because the networks trained on ImageNet dataset [26] already trained the lower level layers (first few layers) on the entire dataset of millions of images which allows them to learn from the various representations of data (e.g. shapes, figures and artifacts). This is consistent with results demonstrated in prior works [33] where transferring features even from distant tasks can be better than using random features. As a result, initializing a network with transferred features from almost any number of layers lead to a boost to generalization that lingers even after fine-tuning the network to the target dataset. This is reported in our Results Table I where a randomly initialized ResNet model (Model A) performs worse than the Baseline Model.

B. Effect of Loss Functions

To estimate the pose accurately, designing an effective loss function is essential. A number of loss functions were tried out and their efficacy is explained. Since the dataset had outliers, MSE loss function was highly sensitive to it and reduced the accuracy. By using L1 loss function, we were able to reduce the error as it's robust towards outliers. It is really challenging to learn both position and orientation simultaneously which uses different units and scales. We found that using a linearly weighted L1 loss, where we weight orientation more than position (Model F) provided better results than having unit weights for both orientation and position (Model A - E). Weighting the loss functions with a scaling factor allows us to effectively balance the losses between position and orientation. This experiment was discussed in Section VI-C3 and the error from using the most effective scaling factor β is reflected in Table I by Model F.

We also noticed that a model which is trained separately to estimate position and orientation had worse accuracy compared to the model which is jointly trained to regress the

position and orientation. For our task we found DarkNet53 architecture (Model H), even though provided better results than Baseline, performed worse than some of the other changes we incorporated.



Fig. 9: Training/Validation Loss vs Epoch Number for Resnet-18 architecture with MSE Loss function

VIII. CONCLUSION

In this project, we successfully used Convolutional Neural Networks for Satellite pose estimation and performed various experiments on the network architecture and loss functions. Some interesting conclusions can be drawn from these experiments, which can be used for improving accuracy in the development of future space navigation systems. First, using transfer learning with Imagenet weights increased the learning rate since Imagenet has understanding of several lower level features. Second, due to large number of training images(12000), using a deeper network (Resnet-50) provided significant improvements over baseline (Resnet-18). Even though Densenet-121 has much deeper network, the learning saturated in our experiments. Also, Densenet architecture is suited for learning multiple features in a vast dataset.

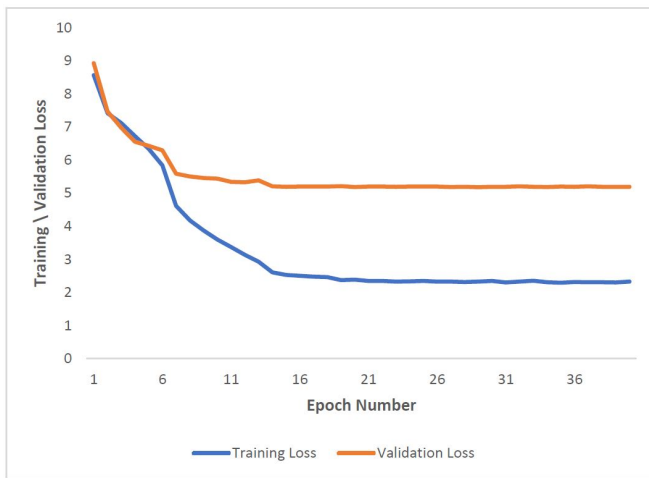


Fig. 10: Training/Validation Loss vs Epoch Number for Resnet-50 architecture with Weighted Loss function

Since our dataset only had satellite images, it didn't improve the accuracy significantly. Additionally, since there are 12000 images in the dataset, training takes a lot of time, approximately 1 hour. We found architecture changes only provided minimal accuracy improvement. But changing the loss function along with the architecture changes provided huge improvements in accuracy (Model F and G). Weighting the loss functions with a scaling factor allows us to balance the losses between position and orientation more effectively. It therefore outperforms the loss functions having unit weights for both orientation and position, as used in Baseline Model and Models A-E.

Potential for future development and enhancements is abundant in this project. First, the networks need to be trained not just using synthetic images but also actual satellite images. In this project, all the training images had satellites behind a black background and earth background which may not be the case always. Sun, comets and other satellites might be present in the background or foreground occluding the target satellite. Thus a much larger and comprehensive dataset is required to train the CNNs. Secondly, the scaling factor used to weight position and orientation in the loss function, requires significant tuning which takes days and it may be possible to incorporate more dynamic loss functions.

Contributions:

Shehzeen Hussain wrote the code to study effect on pose estimation with networks having varying loss functions such as weighted loss functions (Model F), tuning the scaling factors in weighted loss functions to reduce error on unseen Synthetic Image dataset, and the effect on accuracy when using higher number layers in the network (e.g. ResNet-50, Model E). She further experimented with randomly initialized networks and transfer learning with pre-trained networks to study the effectiveness of transfer learning techniques for pose estimation task.

Chinmayee Bhanu wrote the code for the modified branch architecture and the PoseLoss experiments (Model G-H). She experimented with different balance factors and also incorporated the YOLOv3 weights to study the effect of a task specific transfer learning approach compared to generic Imagenet training.

Sneha Kondur performed experiments to study the effect of freezing the lower layer weights and only updating the weights of the last fully connected layers while using transfer learning with ResNet-50 architecture. She also worked on hyper-parameter tuning for batch size and learning rate.

Arunkumar Ravichandran wrote code for estimating pose using Densenet-121 and a modified architecture to estimate position and orientation separately. Tuning was done on Densenet-121 to improve the results compared to baseline Densenet-121 architecture. He also helped with implementing branched architecture and experimenting with both MSE and L1 loss in order to improve accuracy of the models.

All members contributed equally to programming the experiments, debugging code and documenting the results in the final report.

REFERENCES

- [1] A. Cropp, "Pose estimation and relative orbit determination of a nearby target microsatellite using passive imagery." Ph.D. dissertation, University of Surrey (United Kingdom), 2001.
- [2] K. Kanani, A. Petit, E. Marchand, T. Chabot, and B. Gerber, "Vision based navigation for debris removal missions," in *63rd International Astronautical Congress*, 2012.
- [3] S. Sharma *et al.*, "Comparative assessment of techniques for initial pose estimation using monocular vision," *Acta Astronautica*, vol. 123, pp. 435–445, 2016.
- [4] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.
- [5] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004*, 2004.
- [6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [7] A. S. Razavian, J. Sullivan, S. Carlsson, and A. Maki, "Visual instance retrieval with deep convolutional networks," *ITE Transactions on Media Technology and Applications*, vol. 4, no. 3, pp. 251–258, 2016.
- [8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [9] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017.
- [10] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.
- [12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
- [13] M. Benn, "Vision based navigation sensors for spacecraft rendezvous and docking," 2011.

[14] A. Petit, E. Marchand, and K. Kanani, "Vision-based detection and tracking for space navigation in a rendezvous context," in *Int. Symp. on Artificial Intelligence, Robotics and Automation in Space, i-SAIRAS*, 2012.

[15] S. D'Amico, M. Benn, and J. L. Jørgensen, "Pose estimation of an uncooperative spacecraft from actual space imagery," in *5th International Conference on Spacecraft Formation Flying Missions and Technologies*, 2013.

[16] S. Sharma, J. Ventura, and S. D'Amico, "Robust model-based monocular pose initialization for noncooperative spacecraft rendezvous," *Journal of Spacecraft and Rockets*, vol. 55, no. 6, pp. 1414–1429, 2018.

[17] S. Sharma, C. Beierle, and S. D'Amico, "Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks," in *2018 IEEE Aerospace Conference*. IEEE, 2018, pp. 1–12.

[18] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2686–2694.

[19] S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[20] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[21] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[22] H. Sharma, J. Park, D. Mahajan, E. Amaro, J. K. Kim, C. Shao, A. Mishra, and H. Esmailzadeh, "From high-level deep neural models to fpgas," in *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Press, 2016, p. 17.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[24] C. Beierle and S. D'Amico, "Variable-magnification optical stimulator for training and validation of spaceborne vision-based navigation," *Journal of Spacecraft and Rockets*, pp. 1–13, 2019.

[25] M. D'Errico, *Distributed space missions for earth system monitoring*. Springer Science & Business Media, 2013, vol. 31.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[27] "Intuitive guide to convolution neural networks." [Online]. Available: <https://towardsdatascience.com>

[28] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[29] "Coco dataset." [Online]. Available: <http://cocodataset.org/#home>

[30] "Scores for satellite pose estimation challenge." [Online]. Available: <https://kelvins.esa.int/satellite-pose-estimation-challenge/scoring/>

[31] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[32] "L1 loss vs l2 loss." [Online]. Available: <http://risky.github.io/ml/2015/07/28/l1-vs-l2-loss/>

[33] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

IX. APPENDIX A

Here we include a screenshot of the Kaggle Leaderboard where the submission of *whiteranger* is our personal best and in the top-12. We have two other submissions *whiteranger2* and *cbhanu* in the top-15 out of a total of 43 submissions.

Name	Submissions	Last Submission	Best Submission	Real Image Score	Best Score
UniAdelaide	16	June 11, 2019, 12:33 a.m.	June 7, 2019, 12:44 a.m.	4.62639697727509	0.0319533547179452
EPFL_cvlab	22	June 6, 2019, 7:09 a.m.	May 7, 2019, 12:18 a.m.	0.1040193712417902	0.0348374872568371
pedro_fairspace	18	June 9, 2019, 3:52 p.m.	June 9, 2019, 3:52 p.m.	0.14406320273141585	0.0670420745174456
Team_Platypus	15	May 22, 2019, 7:22 p.m.	May 22, 2019, 7:22 p.m.	2.43940309822283	0.102425447526932
motokimura1	19	June 8, 2019, 6:57 p.m.	June 4, 2019, 4:27 p.m.	0.957067212199997	0.116066394386528
Maggies	14	May 4, 2019, 6:45 a.m.	May 2, 2019, 5 p.m.	0.9885750625325125	0.216226125582801
stainsby	14	June 11, 2019, 9:16 p.m.	June 11, 2019, 1:58 p.m.	1.4601098674636728	0.415652616423345
VSI_Feeney	17	June 3, 2019, 3:07 p.m.	May 28, 2019, 7:33 a.m.	1.5748702124396687	0.462855275616921
jblumenkamp	9	June 6, 2019, 4:19 a.m.	June 5, 2019, 1:27 a.m.	4.116387094483612	0.87993156425326
tatima2031	32	June 6, 2019, 10:19 a.m.	June 6, 2019, 10:19 a.m.	1.8301440764310306	1.28309045341051
whiteranger	12	June 12, 2019, 6:22 a.m.	June 12, 2019, 6:22 a.m.	1.621043516109751	1.29120258637829
whiteranger2	10	June 12, 2019, 6:48 a.m.	June 8, 2019, 7:44 p.m.	1.8613352741503806	1.30765830450549
pechora	6	May 23, 2019, 10:43 p.m.	May 21, 2019, noon	1.9986453880821478	1.58482833116868
cbhanu	9	June 4, 2019, 10:01 a.m.	June 4, 2019, 8:39 a.m.	1.8593625982791555	1.72248124250797

Fig. 11: Leaderboard for Kaggle competition - Satellite Pose Estimation <https://kelvins.esa.int/satellite-pose-estimation-challenge/leaderboard/>