Counting Cars in Low-Resolution Overhead Imagery Using a Convolutional Model

Kyle Nguyen
Department of Computer Science
Stanford University
kylen@stanford.edu

Atindra Jha
Department of Computer Science
Stanford University
atj10@stanford.edu

Andrew Conkey
Department of Economics
Stanford University
aconkey@stanford.edu

December 3, 2021

Abstract

Previous work has shown that various convolutional neural network (CNN) and CNN + feature pyramid network (FPN) techniques can achieve high performance on the task of counting cars in high-resolution overhead imagery. However, there is a notable gap in the literature when it comes to counting cars in lower-resolution overhead imagery, which includes most low-cost high-frequency satellite imagery currently available. In this paper, we artificially blur images sourced from the Cars Overhead With Context (COWC) dataset to construct several labelled low-resolution datasets. On these datasets, we train a variety of car-counting models, using the AlexNet, ResNet-18, and ResNet-50 architectures. We observe relatively low out-of-sample performance and significant overfitting in all three models, which we attempt to combat with dropout and $L2$ regularization. Ultimately, our models should be applicable to purposes for which approximate car counts are sufficient, but more work is needed to combat the overfitting issues that we encountered for use-cases where higher accuracies are needed.

1 Introduction

The ability to count cars in high-frequency overhead satellite imagery can significantly enhance the efforts of governments, researchers, and other parties to analyze live trends in traffic, pollution levels, and emissions. Car counting models also hold value for researchers seeking to use car counts as inputs in tackling other problems, such as tracking economic activity [7] or predicting stock prices of retail chains [2, 14].

A number of models have already been developed to count cars from overhead imagery with significant success [1, 3, 7, 12, 16]. However, these models typically rely on high resolution imagery in which cars are clearly distinguishable, typically using satellite resolution in the range of 15-30 cm per-pixel resolution at ground. In contrast, available low-cost high-frequency satellite imagery is of significantly lower resolutions– Planet Labs' daily non-commissioned satellite imagery delivers a 3 m per pixel resolution, while the highest resolution band of NASA's Landsat 8 satellite is only 15 m per pixel [8, 17]. At such resolutions, most cars are represented by only one pixel or less, making even human-level car counting performance extremely poor. While higher-resolution imagery can be commissioned, doing so is extremely expensive and infeasible for those working on a limited budget.

Our project thus focuses on the novel problem of counting cars in progressively lower resolution overhead imagery. To create our datasets, we artificially blur a set of publicly available labelled high-resolution overhead images containing cars. Subsequently, we test a variety of model architectures on the resulting datasets to assess performance and sensitivity to image blurring.

2 Related work

The task of car counting is part of a larger body of work related to small-object detection in satellite imagery. Previous work on the car counting task can generally be divided into two different approaches: 1) convolutional neural network (CNN) based approaches and 2) CNN + feature pyramid network (FPN) approaches. The latter is particularly popular, with many papers [1, 3, 7] working off of a modified RetinaNet [11] with virtually all papers achieving high accuracy on the car counting task. However, while the task of car counting has been approached from many angles for high-resolution satellite imagery and even low-resolution non-satellite imagery [4], we notice that relatively few papers explore the performance of either CNN or CNN + FPN approaches on car counting in lower resolution satellite data, which we explore in this paper.

Our approach to car counting in this paper follows most closely the work by Mundhenk et al. (2016) [12], who also publish the dataset that we use and describe in the *Dataset* section. This paper provides benchmark performance on the car counting task for popular purely-CNN based models such as AlexNet [10] and Inception on this dataset. We replicate the performance metrics reported by Mundhenk et al., but we also note weaknesses regarding the dataset's bias towards 0-label examples and attempt to remedy these in our approach. In addition, we expand further upon previous work by exploring the task of car counting in low-resolution imagery.

3 Dataset and Features

3.1 Dataset

For our dataset, we use high-resolution satellite images of locations with cars from the Cars Overhead with Context (COWC) dataset provided by the Lawrence Livermore National Laboratory (LLNL) [13]. COWC is a large dataset of 32,716 unique annotated cars from the overhead from six distinct locations: Toronto, Canada; Selwyn, New Zealand; Potsdam and Vaihingen, Germany; Ohio and Utah, United States. The images were taken at a satellite resolution of 15 cm, meaning cars are roughly 20 pixels across in the images. Images from Vaihingen and Ohio were only available in grayscale and were therefore omitted from our data set.

Our first task is to replicate the benchmark performance achieved on the COWC dataset by Mundhenk et al. (2016). Following the example of this paper, we begin by writing a Python script to slice all images into 256x256 pixel sub-images yielding a total of 31,437 images.

Each image in the COWC dataset is annotated via a corresponding PNG image containing a single-pixel dot at the center of each car in the image. The slicing ensures that each annotation dot is only contained in one sliced image. However, it does not ensure that the car would not visually be split into multiple images. To avoid this, following Mundhenk et al. (2016), our Python script adds a gray 32-pixel border to each of the sliced images. In addition, we only consider cars whose annotation dots are located at least 8-pixels inside the visible portion of the image. After this process, a total of 15,073 cars are observed across all sliced images.

Location	**# of sliced images**	**Total # of annotated cars**	**Avg. cars per image**
Toronto, CA	3828	4554	1.1897
Selwyn, NZ	13440	559	0.0416
Potsdam, DE	832	778	0.9351
Utah, US	13337	9182	0.6885
Total	**31437**	**15073**	**0.4795**

Of the 31,437 data points, nearly 85% (26,999 data points) were annotated as having 0 cars. Since the data skewed heavily towards images with no cars in them, we undersampled by randomly selecting 1000 of the 26,999 0-annotated images and retaining them, while discarding the rest of the images with 0 cars in them. This resulted in a second dataset with 5,438 images. Note that since we only removed images with 0 cars, we still have a total of 15,073 annotated cars in the undersampled dataset.

Location	# of sliced images	Total # of annotated cars	Avg. cars per image
Toronto, CA	1603	4554	2.8409
Selwyn, NZ	832	559	0.6719
Potsdam, DE	303	778	2.5677
Utah, US	2700	9182	3.4007
Total	**5438**	**15073**	**2.7718**

3.2 Blurring

In addition to the 5,438-image unblurred image dataset, we write a Python script to create two additional datasets of the same images but with a blur factor of 2 or 4. We blur images in two stages. First, we use nearest-neighbor interpolation to reduce an image's size by a constant factor (either 2 or 4). Then, we rescale, again using nearest-neighbor interpolation, to restore the image to its original size.

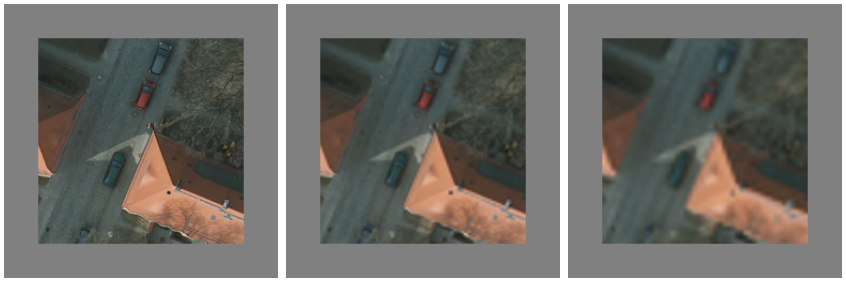


Figure 1. A typical training example unblurred, 2x blurred, and 4x blurred, left to right.

For the blurred images, we then use the same slicing and border process as for the unblurred images. The 32-pixel gray border is applied after the blurring process and therefore remains unblurred. After image processing, we are left with a total of 4 datasets of sliced and framed images:

1. The initial dataset with 31,437 unblurred images.
2. The undersampled dataset with 5,438 unblurred images.
3. The 5,438-image dataset blurred by a factor of 2 (low-blur).
4. The 5,438-image dataset blurred by a factor of 4 (high-blur).

4 Methods

We begin by training a few baseline models on the original dataset of 31,437 unblurred images to compare benchmark performance on the car counting task. We implement all of the following models using PyTorch [15]. For the baseline model, we use transfer learning on the AlexNet [10] architecture trained on ImageNet. We replace AlexNet's original 1000-class softmax output with a 50-class softmax output, where each class corresponds to a discrete car count. The maximum number of cars that can be observed in a single training image is 47. Inspired by the model of Mundhenk et al. (2016), we round this up and use 50 output classes to allow for the possibility of generalizing to future data that may include more than 47 cars.

In addition to AlexNet, we train a ResNet18 and ResNet50 [5] with the same setup. For all architectures, learning is conducted via an Adam optimizer [9] for 50 epochs with a batch size of 64. Since we use a softmax output layer and essentially treat this as a 50-class classification problem, we use cross-entropy loss to minimize the distance between the predicted probability and the true class label.

$$L(y, \hat{y}) = -\frac{1}{m}\sum_{i=1}^{m} y_i log(\hat{y}_i)$$

Formula 1. Cross-entropy loss formula. y_i is a 50-dimensional one-hot vector representing the true car count in an image.

We use a weight-decay parameter of 0.0001 and a learning rate of 0.0001 as these are commonly-cited values in computer vision literature. For all training, we use a 80% train, 10% dev, 10% test split.

As discussed later in *Results*, we achieve high accuracy and low loss numbers, close to what is reported by Mundhenk et al. (2016), on all models using this dataset. However, as mentioned in the *Dataset* section of this paper, we find that nearly 85% of the 31,437 data points were annotated as having 0 cars. We suggest that this imbalance of car counts in the dataset artificially boosted the performance of our models as well as possibly of the models used in Mundhenk et al. (2016). Since models try to reduce classification errors across the entire dataset, it is preferable for the training set to not have high class imbalance. Therefore, we train, validate, and test all three of our aforementioned models independently using the new undersampled dataset with 5,438 unblurred images.

All three of our models overfit the undersampled training set to differing degrees, resulting in high variance. We therefore attempt to enhance the regularization of our best-performing model, ResNet18 with a 50-class softmax output by 1) adding dropout with a p of 0.5 right before the fully connected layer; and 2) testing a range of values for the weight decay parameter of the Adam optimizer in order to gradually increase L-2 regularization. Specifically, we test values between 10^{-2} to 10^{-4}, incrementing the exponent by intervals of 0.25. Our results indicate that $10^{-2.5}$ is the optimal weight decay value for our model for counting cars in unblurred images based on dev set performance.

Moving on to blurred images, we train, validate, and test our two blurred image datasets (2x blur as well as 4x blur) with 5,438 data points each using our modified ResNet18 convolutional neural network (with dropout and a 50-class output layer). We, again, test a range of different weight decay values across different levels of blurring to ensure that our model has been fine-tuned to perform optimally on the low-blur as well as the high-blur car counting tasks. We note that the model performed best on 2x blurred overhead imagery with a weight decay value of $10^{-2.75}$ and on the 4x blurred imagery with a weight decay value of $10^{-3.75}$. It makes intuitive sense that prediction on lower-resolution imagery would be most accurate with a smaller weight decay hyperparameter, since blurred images contain fewer features which itself reduces overfitting.

5 Results and Discussion

As noted Table 1, we achieve very high performance for each of our three models with the initial dataset of 31,437 unblurred images. The numbers are in line with the performance of the models mentioned in Mundhenk et al. (2016). However, we believe that the high-performance numbers achieved by Mundhenk et al. (2016) as well as by our models were a result of the high class imbalance (heavily favouring 0-label examples) in the dataset, leading to significantly worse average car-count precision during classification of images with one or more cars in them.

Table 1. Test performance metrics of AlexNet, ResNet18, and ResNet50 on the 31,437 unblurred image dataset. MAE is Mean Absolute Error. RMSE is Root Mean Squared Error. Proposal is the accuracy with which a model classifies an image as having at least one car or not.

Model	Accuracy	Accuracy (±1)	Accuracy (±2)	Average Loss	MAE	RMSE	Proposal
AlexNet	89.4%	96.0%	97.7%	0.005546	0.204517	0.890828	94.9%
ResNet18	91.3%	96.9%	98.4%	0.004635	0.178753	1.046319	96.5%
ResNet50	90.9%	97.1%	98.6%	0.004176	0.156170	0.677942	97.3%

Our downsampled dataset of 5,438 unblurred images gives us the following results (Table 2) when run on the three baseline models of AlexNet, ResNet18, ResNet50.

Table 2. Test performance metrics on the 5,438 unblurred image dataset.

Model	Accuracy	Accuracy (±1)	Accuracy (±2)	Average Loss	MAE	RMSE	Proposal
AlexNet	41.5%	79.8%	89.7%	0.037740	1.090074	2.278012	94.9%
ResNet18	55.9%	84.9%	93.2%	0.032278	0.750000	1.452685	96.3%
ResNet50	57.9%	82.4%	91.4%	0.043241	0.889706	2.065116	92.5

These results are more representative of the actual performance of the models since the classes are relatively less imbalanced in our modified undersampled data set. As expected, we observe relatively higher losses and lower accuracies across the board as a consequence of our data set modification, since the task becomes

harder with the skew towards 0-label images removed. The differences between the performance of each model were also more pronounced thanks to the aforementioned resampling. We observe high overfitting in our models. To remedy this, we choose to further modify our ResNet18 to add a dropout layer and try out a range of weight decay values to achieve optimal performance. Key performance metrics of the ResNet18 with dropout and various weight decay values on the unblurred, low-blur, and high-blur datasets can be seen in Appendix 8.3.1–3. A summary of the best-performing models (as selected by comparing exact accuracy on the dev set) can be seen below.

Table 3. Test performance of the ResNet18 with Dropout regularization added, selected by the best-performing weight decay (WD) parameter of the Adam optimizer, across all blur levels.

Dataset	Optimal WD	Accuracy	Accuracy (±1)	Accuracy (±2)	Average Loss	MAE	RMSE	Proposal
Unblurred	$10^{-2.5}$	60.1%	86.0%	93.9%	0.021848	0.806985	2.094724	96.3%
Low-blur (x2)	$10^{-2.75}$	57.7%	82.0%	91.9%	0.030398	0.900735	2.194579	93.9%
High-blur (x4)	$10^{-3.75}$	56.6%	84.9%	92.8%	0.035743	0.790441	1.735232	94.7%

With a combination of dropout regularization and fine-tuning the weight decay parameter, we were able to achieve slightly better test performance on the unblurred dataset (60.1% accuracy as compared to 55.9% accuracy in the unregularized model using 10^{-4} weight decay and no dropout). As expected, even when selecting the best model, accuracy falls as blur level increases, though even at the highest blur level, we were still able to achieve 56.6% exact accuracy. All models achieved ±1 accuracy $> 80\%$ and ±2 accuracy $> 90\%$ after fine-tuning.

6 Conclusion and Future Work

Unfortunately, we encountered significant overfitting issues in the training of our models. While introducing dropout and more severe weight decay lead to improvements in out-of-sample performance, the resulting models were still not sufficiently accurate to reach human-level performance on calculating exact car counts. However, our ±1 and ±2 accuracy rates are strong, meaning that our models still hold value in contexts where exact prediction is unnecessary. In use-cases such as traffic or economic activity prediction, which involve counting cars in contexts in which many are present, less than perfect accuracy is not a significant issue.

To attempt to further reduce overfitting, future work might involve running less complex models (such as an even shallower ResNet). Some past work was also able to achieve much higher accuracy by first training an object localization model and then using those results to calculate counts [1]. We chose not to take this approach, thinking it would be more effective (especially in the blurred datasets) to directly estimate our variable of interest and hoping to avoid severe computational constraints. However, changing to an object localization approach would likely significantly reduce overfitting, since object localization output is more complex than object counting output. Due to the performance issues we encountered even in the high resolution case, we did not move onto even lower resolutions such as 16x blurring, which would have more accurately reflected the type of images found in, for example, the Planet Labs dataset [8].

Had we successfully trained a model with near-perfect accuracy on high-resolution imagery, we would have used it to label an external, unlabelled dataset of parking lot satellite images that we scraped from OpenStreetMap and Google Maps (see code). This would have allowed us to significantly expand our training set for the blurred image models without need for manual labelling by blurring the newly labelled, larger dataset. This data augmentation technique was at the core of our initial strategy, and we still believe it would be an effective way to improve future models seeking to predict car counts from lower-resolution imagery.

7 Contributions

All group members were involved in driving the overall direction of the project. Each group member conducted literature review and contributed to enhancing the models. Further specific responsibilities are outlined below.
Kyle

- Python scripting for data processing (slicing, blurring)
- General Pytorch scripting for model optimization
- AWS environment setup and cloud management

Atindra

- Pytorch scripting, implementing the softmax classifier skeleton and train/test loop functionality.
- Implementing pretrained AlexNet/ResNet models
- Implementing regularization techniques (Dropout, weight decay)

Andrew

- Portion of blurring code
- Sequential looping for model training
- Hyperparameter optimization

References

[1] Arthur Douillard. "Detecting Cars from Aerial Imagery for the NATO Innovation Challenge." *Arthur Douillard*, 21 June 2018, https://arthurdouillard.com/post/nato-challenge/.

[2] Counts, Laura. "How Hedge Funds Use Satellite Images to Beat Wall Street-and Main Street: Haas News: Berkeley Haas." *Haas News | Berkeley Haas*, 21 Nov. 2019, https://newsroom.haas.berkeley.edu/how-hedge-funds-use-satellite-images-to-beat-wall-street-and-main-street/.

[3] Gao, Peng, et al. "Vehicle Detection with Bottom Enhanced RetinaNet in Aerial Images." *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, 2020, https://doi.org/10.1109/igarss39084.2020.9323216.

[4] Hardjono, Benny, et al. "Vehicle Counting Evaluation on Low-Resolution Images Using Software Tools." *Proceedings of the 9th International Conference on Information Communication and Management*, 2019, https://doi.org/10.1145/3357419.3357453.

[5] He, Kaiming, et al. "Deep Residual Learning for Image Recognition." *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, https://doi.org/10.1109/cvpr.2016.90.

[6] Hendrycks, Dan, et al. "Using Pre-Training Can Improve Model Robustness and Uncertainty." *ArXiv.org*, 20 Oct. 2019, https://arxiv.org/abs/1901.09960.

[7] Hill, Cole. "Automatic Detection of Vehicles in Satellite Images for Economic Monitoring." *Digital Commons @ University of South Florida*, https://digitalcommons.usf.edu/etd/8792/.

[8] "Homepage." *Planet*, https://www.planet.com/.

[9] Kingma, Diederik P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization." *ArXiv.org*, 30 Jan. 2017, https://arxiv.org/abs/1412.6980.

[10] Krizhevsky, Alex, et al. "ImageNet Classification with Deep Convolutional Neural Networks." *Communications of the ACM*, vol. 60, no. 6, 2017, pp. 84–90., https://doi.org/10.1145/3065386.

[11] Lin, Tsung-Yi, et al. "Focal Loss for Dense Object Detection." *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, https://doi.org/10.1109/iccv.2017.324.

[12] Mundhenk, T. Nathan, et al. "A Large Contextual Dataset for Classification, Detection and Counting of Cars with Deep Learning." *Computer Vision – ECCV 2016*, 2016, pp. 785–800., https://doi.org/10.1007/978-3-319-46487-9_48.

[13] Mundhenk, T. Nathan. "Cars Overhead with Context Dataset at LLNL." *gdo152.Llnl.gov*, https://gdo152.llnl.gov/cowc/.

[14] Partnoy, Frank. "Stock Picks from Space." *The Atlantic*, Atlantic Media Company, 23 Apr. 2019, https://www.theatlantic.com/magazine/archive/2019/05/stock-value-satellite-images-investing/586009/.

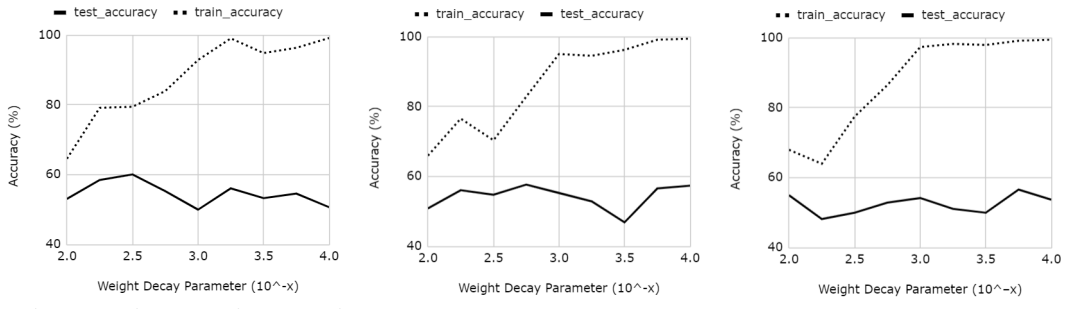
[15] Paszke, Adam, et al. "Pytorch: An Imperative Style, High-Performance Deep Learning Library." *ArXiv.org*, 3 Dec. 2019, https://arxiv.org/abs/1912.01703.

[16] Tayara, Hilal, et al. "Vehicle Detection and Counting in High-Resolution Aerial Images Using Convolutional Regression Neural Network." *IEEE Access*, vol. 6, 2018, pp. 2220–2230., https://doi.org/10.1109/access.2017.2782260.

[17] *What Are the Band Designations for the Landsat Satellites?*, https://www.usgs.gov/faqs/what-are-band-designations-landsat-satellites?qt-news_science_products=0qt-news_science_products.

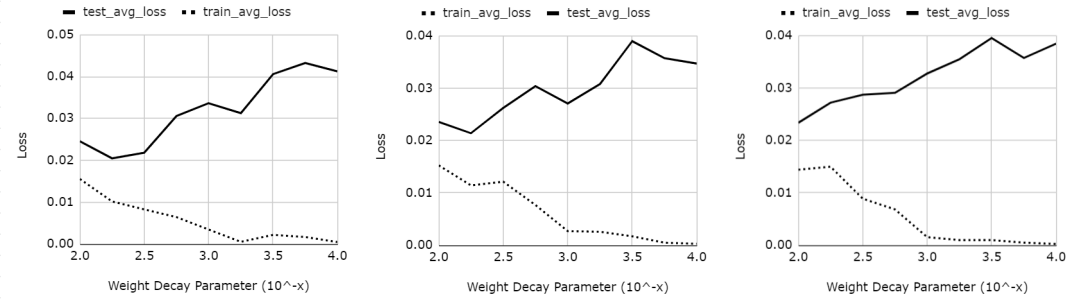
8 Appendix

8.1 Classification accuracy across varying weight decay values



From left to right, these graphs show (exact) accuracy on the unblurred, 2x blurred, and 4x blurred datasets across different weight decay parameter values. We observe a slight decay in performance as blurring increases.

8.2 Average loss across weight decay values



From left to right, these graphs show average loss on the unblurred, 2x blurred, and 4x blurred datasets across different weight decay parameter values. We observe a slight decay in performance as blurring increases.

8.3 Overall test performance metrics

8.3.1 Unblurred dataset

weight_decay	test_accuracy	test_within_one	test_within_two	test_avg_loss	test_mae	test_rmse	test_proposal
2	53.1	82.9	92.5	0.02458	0.9375	2.368606	93
2.25	58.5	**87.5**	**96.1**	0.020512	**0.698529**	1.8301	**96.3**
2.5	**60.1**	86	93.9	0.021848	0.806985	2.094724	96.3
2.75	55.3	82.7	90.1	0.030635	0.873162	1.845603	95.2
3	50	80	89.3	0.033674	1.0625	2.241815	92.3
3.25	56.1	83.5	92.6	0.031304	0.759191	**1.426508**	94.9
3.5	53.3	80.3	90.6	0.040612	0.895221	1.763081	93.2
3.75	54.6	82.9	92.3	0.043288	0.856618	1.781234	95.8
4	50.7	82	93	0.041268	0.862132	1.623019	93

8.3.2 Low-blur (x2) dataset

weight_decay	test_accuracy	test_within_one	test_within_two	test_avg_loss	test_mae	test_rmse	test_proposal
2	50.9	83.5	91	0.02359	0.959559	2.114377	**97.4**
2.25	56.1	**86.6**	**94.1**	0.021425	**0.704044**	**1.330496**	95.8
2.5	54.8	82.4	93.6	0.02625	0.895221	2.092089	91.4
2.75	**57.7**	82	91.9	0.030398	0.900735	2.194579	93.9
3	55.3	84	93	0.027097	0.78125	1.684161	96.5
3.25	52.9	80.3	92.1	0.03081	0.84375	1.521295	93.2
3.5	46.9	77.4	87.5	0.038996	1.038603	1.806341	93.9
3.75	56.6	84.9	92.8	0.035743	0.790441	1.735232	94.7
4	57.4	84.6	92.3	0.03473	0.742647	1.429726	93.8

8.3.3 High-blur (x4) dataset

weight_decay	test_accuracy	test_within_one	test_within_two	test_avg_loss	test_mae	test_rmse	test_proposal
2	55	82.7	90.8	0.023379	1.049632	2.650957	94.5
2.25	48.2	80.9	91.9	0.02722	0.933824	1.866894	94.3
2.5	50	78.7	88.8	0.028737	1.086397	2.278819	**96.3**
2.75	52.9	84	92.6	0.029101	0.810662	**1.582882**	92.5
3	54.2	84.4	**93.6**	0.032786	0.806985	1.744213	93.4
3.25	51.1	79.8	90.6	0.03549	0.946691	1.815477	95.6
3.5	50	78.9	89.9	0.039551	1.090074	2.576406	92.6
3.75	**56.6**	**84.9**	92.8	0.035743	**0.790441**	1.735232	94.7
4	53.7	81.4	91.2	0.038466	0.920956	2.105228	95.2