ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Ευφυής Αναγνώριση Καταστροφών σε Κτίρια

Γρηγορόπουλος Νικόλαος

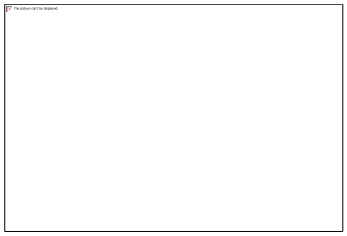
ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κολομβάτσος Κωνσταντίνος

Επίκουρος Καθηγητής

Λαμία 2022



ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Ευφυής Αναγνώριση Καταστροφών σε Κτίρια

Γρηγορόπουλος Νικόλαος

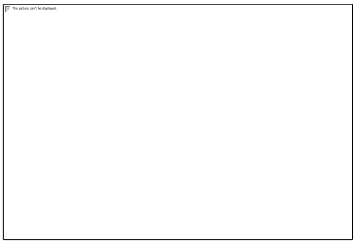
ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κολομβάτσος Κωνσταντίνος

Επίκουρος Καθηγητής

Λαμία 2022



SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

Damage assessment in Buildings with the help of Deep Learning

NIKOLAOS GRIGOROPOULOS

FINAL THESIS

ADVISOR

Dr Konstantinos (Kostas) Kolomvatsos, BSc MSc PhD

Assistant Professor in Intelligent Systems for Pervasive Computing

Lamia ……2022

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [1], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ.), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: ……/…../2022

Ο Δηλών.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση

του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων

σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Η αυτοματοποίηση της εκτίμησης ζημιών σε κτίρια μπορεί να είναι δύσκολη. Σε αυτή την εργασία θα δούμε πώς η εφαρμογή αρχών μηχανικής μάθησης και μοντέλων βαθιάς μάθησης διευκολύνει κατά πολύ την επίλυση τέτοιων προβλημάτων. Θα διαπιστώσουμε ότι η μηχανές υποστήριξης διανυσμάτων (Support Vector Machines – SVMs) μπορούν να έχουν σημαντικό αντίκτυπο στο στάδιο ταξινόμησης ζημιών και να μας δώσουν πιο ακριβή αποτελέσματα. Το μοντέλο μας χρησιμοποιεί δύο διαφορετικά νευρωνικά δίκτυα, ένα δίκτυο για ταξινόμηση κτιρίων και ένα δεύτερο με SVM λογική ως τελική αναγνώριση που χρησιμοποιεί τα βάρη του πρώτου για να ανιχνεύσει κτίρια και να εκπαιδευτεί στην εκτίμηση ζημιών με βάση ένα σύνολο δεδομένων δορυφορικής εικόνας που απεικονίζουν το πριν και το μετά. Κάνουμε μια σύγκριση μεταξύ του SVM και του μοντέλου που δεν βασίζεται σε SVM για να δούμε αν το επίπεδο SVM έχει κάποιο αντίκτυπο στην αναγνώριση καταστροφών.

ABSTRACT

Automating damage assessment in buildings can be a challenging research topic. In this thesis, we are going to see how the application of machine learning algorithms and deep learning models can facilitate towards finding the solution of this problem. We elaborate on if Support Vector Machines (SVM) can have any major impact in the damage classification phase and provide us with more accurate results. Our model adopts two different neural networks, one for building the classification model and a second one with involving the SVM as the last layer that uses the weights of the first network to detect buildings and perform training on damage assessment based on a pre/post satellite image dataset. We do a comparison between the SVM and non SVM based model to see if the SVM layer has any impact to the damage classification.

Table of Contents

Introduction

It would benefit our society, if we could know the damage assessment of buildings and therefore trapped humans in them after a natural disaster in a certain area (e.g., the capital city of Ukraine) in a few hours. In a natural disaster scenario (e.g., earthquake, tsunami) where affected areas are inaccessible (e.g., flooded) by land we need to know how many buildings have been damaged. Something like this can be achieved by grading the buildings through human visual inspection. An approach like this provides the most reliable classification method but needs proper staff training prior to the occurrence of the disaster and plenty of labor hours which translates into money and since we are dealing with a disaster the most critical of all, time. So, by automating the whole classification process rescuers could save time to focus on rescuing trapped people (Xu, Lu, Li, Khaitan, & Zaytseva, 2019) [1].

Over the last decade there have been huge advancements in the field of Machine Learning and consequently in deep neural networks (DNNs).

We are going to use SpaceNet to train our model and detect what is a building in every satellite image. Then, using ResNet50 with some extra conv layers with the last of them being an SVM classifier we will train our model to identify level of damages in buildings and compare that to the baseline approach that xview2 provides to see if SVM as final layer can be effective.

The main contribution of this paper is the study of machine learning along with SVMs in situations like humanitarian crisis and damage estimation after a major natural disaster.

To properly train a model like the proposed, someone can train the model to recognize what a building is and then using satellite images of pre/post disaster images train it to the level of damage of any given image.

Machine Learning (Classifiers, SVMs, DMTs)

We take advantage of machine learning in our everyday lives, sometimes without even noticing. From chatbots, language translation, text editors and even Amazon similar product recommendations. Most of the companies nowadays use the term Artificial Intelligence and machine learning as homonyms. Actually, the second one is a subcategory that gives computers the ability to learn things without having to be programmed to do a specific task. Even the companies that have not adopted Machine Learning yet are planning to use it in the coming years." It is important to engage and begin to understand these tools, and then think about how you are going to use them well. We must use these [tools] for the good of everybody" said Dr. Joan LaRovere1. Considering that we are going to analyze the different machine learning approaches along with various classifiers (e.g., Support Vector Machines - SVMs, Decision Making Trees) and see how the study of SVM and damage assessment can help the society address natural disasters more efficiently.

Therefore, supervised machine learning models needs to be trained using labelled data sets which allow the model to learn and grown over time by itself. For example, providing the model with images of buildings labelled by humans, the model can learn what a building is and classify images containing buildings without prior access to them.

On the other side, unsupervised machine learning can look for patterns in un-labelled data to find patterns or the trends. A model like that can look through a company's sales data and identify several types of clients and therefore the company can provide more targeted ads to each of them.

Finally, we have reinforcement machine learning which trains the model through trial and error and reward the model when it makes the best decision. Models like this can and have been used numerous times to train Stockfish [15] or Othello 8 x 8 and even autonomous vehicles so they will know that they made the right decisions and by doing that they will learn over time what actions they should make [16].

1 https://www.childrenshospital.org/directory/physicians/l/joan-larovere

Figure

0 Thomas Malone, MIT Sloan See: https://bit.ly/3gvRho2

Machine Learning with SVM

According to the OpenCV documentation the definition of a SVM is "a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples" [4].

The SVM algorithm, in its original form, is an algorithm for two-class classification, meaning that the algorithm, in its infancy, could only classify data into two distinct classes. It maps the input vectors into a high dimensional feature space. In this feature space, a hyperplane is created with properties that ensure a high generalization ability of the network (Cortes & Vapnik, 1995). The optimal separating hyperplane discriminates the data set fifteen into discrete number of classes that minimizes misclassification attained during the training phase (Kavzoglu & Colkesen, 2009) (Maulik & Chakraborty, 2017).The implementation of a linear SVM assumes that the feature data are linearly separable. However, in practice, data points of different classes often overlap with each other making the basic linear decision boundaries insufficient. Kernel functions have thus been developed (Mountrakis, Im, & Ogole, 2011).Non-linear kernels, like the radial-basis function, map the input to a higher spatial dimensional feature space where a linear decision boundary, via hyperplane, separates the classes. SVMs can manage small training data sets while often producing higher classification accuracy than traditional methods (Mantero, Moser, & Serpico, 2005). In addition, the SVM learning process follows a structural risk minimization scheme that minimizes classification error on unseen data without any previous assumptions made on the probability distribution of the data. Another benefit of the SVM model is its ability to strike a balance between accuracy on a limited amount of training data and the ability to generalize unseen data (Mountrakis, Im, & Ogole, 2011).This means the SVM can strike a reasonable balance between bias and variance. If the classifier has too many adjustable parameters (bias), it will learn the training data without difficulty, but it is unlikely that

it will generalize properly for the unseen data patterns (Boser, Guyon, & Vapnik, 1992).One challenge of the SVM model is the choice of an optimal kernel. Although there are many kernel functions, some are not optimal for certain remote sensing applications (Mountrakis, Im, & Ogole, 2011). Research has shown that kernels such as the radial basis function and polynomial kernels produce different results when applied to satellite imagery (Zhu & Blumberg, 2002).

Linear SVM

It is the simplest form of Support Vector Machine that does not take advantage of kernels and sometimes also called Large Margin Classifier so it can only classify data into two distinct classes. In a random plot that has two classes (binary classification) we separate them in many ways (vertical lines that split data in half). SVM classify samples to split the data with the largest margin possible. So, a Support vector is either close to the boundary or falsely classified. So basically, SVM tries to find the optimal hyperplane that split the data into with the highest maximized margin. When we train a SVM model support vectors have a major impact in the classification process, that by removing nonsupport vector (far away from the boundary samples) has no impact to the model at all.

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x_1}}$$

The cost function that is used to train the SVM is remarkably like Logistic Regression function but with a piecewise linear approach. Achieving lower theta values, we can be more confident that our result is more accurate.

Non-Linear SVM

Nonlinear SVM is used to classify non-linear separated data and therefore someone cannot classify it using a straight line to separate into two categories. The classifier is called Non-Linear SVM Classifier. When we are dealing with non-linear data, we need to separate our data in a way that using a straight line cannot help us. For example, in three dimensions using a circle to encapsulate one area provides us with a binary classification in the three dimensions by assuming everything outside of this circle belongs to one class and vice versa.

Deep Learning

Imagine what is like being an infant and keep asking if every individual item that you stumble across is a dog. After many iterations of the same question the infant will learn to recognize the characteristics that every dog has and therefore recognize them easier. Similarly Deep Learning use neural networks with many layers to extract if the given image is a dog or not. To achieve this behavior the network, learn what a dog is using a labelled dataset and extract features that every dog has (e.g., tail, fur, four legs etc.) so when you give the model a new dog image it can use some layers to detect individual features of a dog (e.g., paws, tail, mammary glands etc.) while another layer will conclude if all this assemble a dog or not. Sometimes those models are so smart that they can even do breed classification along with whether if the image contains a dog or not.

ResNet

Deep convolution neural networks are widely used for image classification. Deep Networks contain level features and classifiers that can be benefit by stacking more layers to extend the depth of the network. Using more than usual layers in deep learning models is a quite common approach in visual recognition tasks (e.g., against ImageNet dataset) alongside with some non-visual problems but how are we sure that stacking more layers is going to give as a better model performance? The first part of the equation to this problem was the problem of vanishing/exploding gradients (Bengio, Simard, & Frasconi,

1994)[10] but using normalized initialization and intermediate normalization layers gave the networks with tens of layers the ability to start converging for stochastic gradient descent with back propagation. However, after applying the new layers the networks training gets saturated and starts to degrade leading to a degradation problem that cannot be fixed by providing more layers because that will only give the network higher training errors. So, to fix the degradation problem the authors of (He, Zhang, Ren, & Sun, 2015)[3] proposed a deep residual learning framework.

A residual neural network (ResNet) [3] is a special type of artificial neural network (ANN) that has been proposed by (He, Zhang, Ren, & Sun, 2015) [3] and won the first place in the ILSVRC 2015 with a top5-error of 3.57%. They were inspired by VGG and created a plain network with same time complexity across the layers by keeping or doubling the filters based on the output of the feature map and down sampling to convolutional layers with a stride of two. The last layer used global average pooling layer and a 1000-way fully connected layer with SoftMax providing thirty-four weighted layers way less that VGG nets. Based on the plain network they created shortcut connections to achieve the residual network.

ResNet tend to get higher accuracy for increased depths compared to other approaches.

ResNet50

ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer.

The architecture of ResNet in general and therefore ResNet50 is:

Inceptionv3

Inception-v3 is a convolutional neural network architecture from the Inception family that makes several improvements including using Label Smoothing, Factorized 7 x 7 convolutions, and the use of an auxiliary classifier to propagate label information lower down the network (along with the use of batch normalization for layers in the sidehead).

Detection of Damages in Building

As the problem of detecting damages in buildings is quite extensive and the approaches to solve it vary from mathematical prediction models based on previous natural disasters data. Specifically for an earthquake disaster the observation of seismic activity before an earthquake occurs helps the calculation of the impact of damage in buildings .Therefore this will help rescuers to decide which building are more crucial for human evacuation (Morales-Valdez, Alvarez-Icaza, & Escobar, 2020).In the other hand we have properly trained deep learning models (e.g., ResNet) with the help of pre/post disaster satellite images for real-time classification damage assessment. More specifically on earthquake impacted buildings that human inspection is difficult, we can benefit from the use of an Unmanned Aerial Vehicle (UAV) that can go to the affected area relatively easier and provide the rescuers with a damage assessment report for every single impacted building.

Building Damage Assessment Using Deep Learning and Ground-Level Image Data

With the help of computer vision, Karoon Rashedi and Nia Greg Mori [5] tried to solve a different variation of the problem that was mentioned in the introduction. Their main difference is that this approach performs damage classification on buildings when the remote sensor is on ground-level instead of aerial image input. By using only post-disaster images from a small dataset that they created on their own as of the time of publishing the paper there was no prior properly labeled dataset with round-level images of areas affected to fit with their approach. So only by using post image data of damaged or not-damaged building they implemented three different convolutional neural networks each of them designed to perform an extremely specific task. The first one simply analyzes the image. Both, second and third, require some image preprocessing because images sometimes contain irrelevant information in comparison to buildings, such as roads, cars and basically anything that is not a building so to fix that they used a semantic segmentation algorithm to extract buildings and anything else in the image that indicates that buildings have any damage in them (e.g., broken walls on the ground close to the house). The Convolution Networks later take the raw and preprocessed data in order to extract features and as final step a regressor is used to output a percentage value corresponding to the damage that the image has. They only use post images, and the output is a continuous value as a factor measurement rather than predefined damage categories. (Nia & Mori, 2017) [5]

Earthquake Damage Assessment Based on Deep Learning Method Using VHR Images

Another approach of the damage assessment problem and human security after a natural disaster was addressed with the help of Deep Learning Neural Networks along with VHR (Very High Resolution) satellite images by Masoud Moradi and Reza Shah-Hosseini and presented at the 3rd International Electronic Conference on Geosciences. They used Haiti earthquake that happed on the capital city Port-au-Prince on 12 January 2010 as their base rural area to build their training dataset. Pre disaster images of the area acquired after four days of the earthquake and post disaster are based on images from 1 October 2009. Both pre/post images had four multi spectral bands along with one high resolution

band. To proper label the above-mentioned approach of dataset the International Institute UNITAR / UNOSAT data and Earthquake Geo-spatial Data “Dataverse” (CGA, Harvard University) was used. After the collection of pre/post images they compiled every pre/post VHR image into a large image with every pixel have a binary value that represent the state of the destruction of the building structure. The newly created images were projected into UTM/WGS84 geo-referenced coordinate system, and they used random patches of the dataset that have higher that fifty percent of pixel labeled as damaged or undamaged. Their base network architecture and training is based on the proposed of UNet (Ronneberger, Fischer, & Brox, 2015)[6] along with Deep Residual UNet (Zhang, Liu, & Wang, 2018) [7]. To accelerate the network convergence, they took advantage of Batch Normalization and normalized the input layer by performing adjusting and scaling across the activation layers of the network. Convolution layer was used to replace the max pooling layer that all UNet have by nature because it performs better. In addition, they used a batch size of twenty-five along with image patch size 256x256 used to train the UNet model for about fifty epochs with a learning rate of 0.01 along them. Root Mean Square Probability was used as the parameter optimization as we speak for a large dataset and cross-entropy as the loss/cost function. The model they proposed is used for mapping earthquake building condition assessment but with the proper dataset for any other type of natural disaster (e.g., fire, tsunami, etc.) and the corresponding labeling it can be trained again as it is a supervised model and well developed so it can meet the criteria of any other natural disaster easily (Moradi & Shah-Hosseini, 2020)[8].

Our Model

Firstly, before starting to explain our deep learning model, we must first collect data for the problem we are willing to solve because the more data someone has for a problem the better the model is going to be trained. This can be done quite easily with the help of DIU (Defense Innovation Unit**)** that offers through the xView2 challenge site the xBD (Gupta, et al., 2019) [2], a large-scale dataset containing pre/post satellite imagery with ground sample distance lower than 0.8 for various natural disasters (e.g., earthquakes, wildfires, etc.) covering plenty of geographical locations along with over 800,000 building annotations. Also due to that wide building coverage across many regions the dataset creates a perfect diversity for different building (e.g., sizes, techniques, etc.) along with negative satellite images that contains areas with undamaged buildings or no buildings at all making the perfect for checking if building classification was trained properly. After consulting disaster response experts they created a joint damage scale that properly representes real building damage conditions as shown in Figure 1.

Figure 1 Joint Damage Scale from xView2 site

Model

"The localization model was based on a SpaceNet submission by Motoki Kimura, which featured an altered U-Net architecture [18]. We lightly modified this model to fit our dataset. The model was trained on an eight GPU cluster for seven days. The model achieved an Intersection over Union of 0.97 and 0.66 for "background" and "building," respectively.

The classification model is shown in Figure 2. The ResNet50 is pre-trained on ImageNet [4] whereas the smaller side network is initialized with random weights. All convolutional layers use a ReLU activation except the last one that uses soft-max activation. The output is a one-hot encoded vector where each element represents the probability of an ordinal class. The model uses an ordinal cross-entropy loss function. Unlike traditional cross-entropy, ordinal cross-entropy penalizes relative to the distance between true and predicted ordinal class. Since the difference between any two classes is not

interchangeable, this loss function allows the model to better distinguish between the various levels of damage."

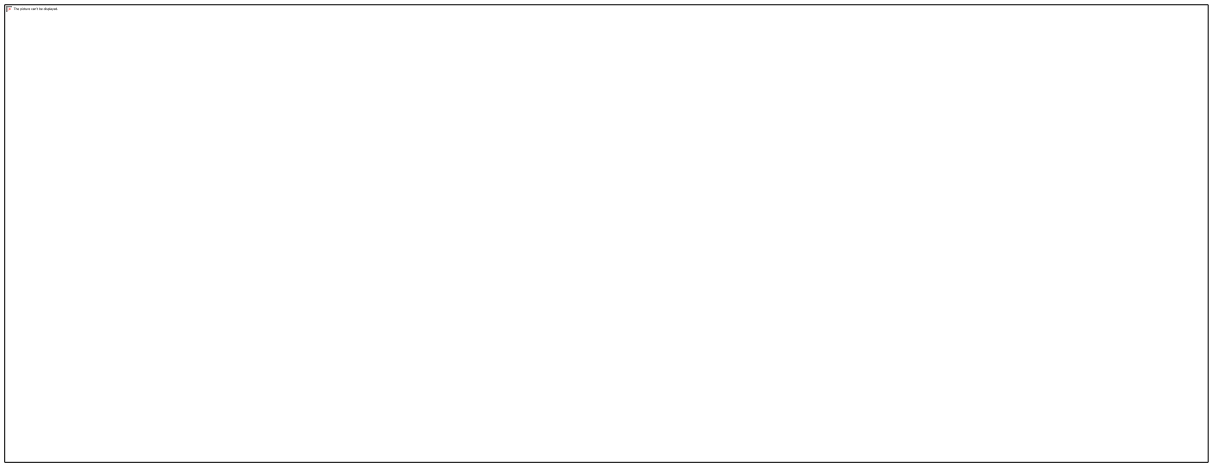


Figure 2 Architecture of the baseline classification model. The input is fed into a pre-trained ResNet 50 as well as a shallow CNN. The outputs of each stream are concatenated and passed into dense layers for classification

Uses the Space Net [9] model for the building classification that is based in UNet a Convolutional Network for Biomedical Image Segmentation [6] and a damage training classification based on ResNet50 with some additional "fuzz" layers and for the last layer we simulate an SVM using categoral_hinge as the loss function to determine in which of four classes of the damage scale each individual building belongs to.

Use cases and performance results.

Taking into account the earthquakes that happened in October 2021 in Heraklion, Crete with magnitudes varying from small ones to 6,3 in the scale of Richter as seen in Figure 3 and in general Greece having a high seismic activity though out its land we believe that a deep learning model like this can be a very powerful "weapon" in the hands of civil

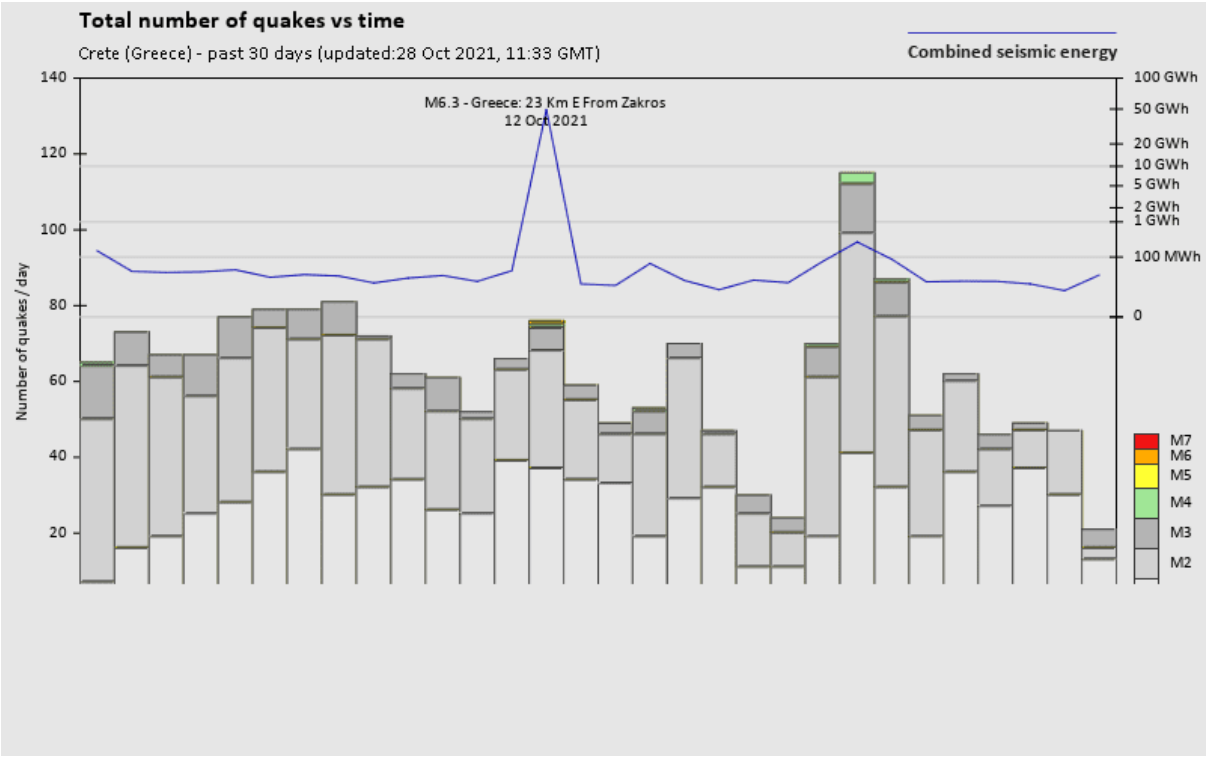


Figure 3 Total Number of Earthquakes in Crete

protection rescuers to properly have an estimate of damages in the affected area and more efficiently organize the teams to go to buildings more targeted rather that visually inspecting each one and deciding on premise. Also sometimes given the high number of Richter some places become obscured and inaccessible by foot so having the damage assessment for each building can help provide aerial support to these buildings and therefore rescue trapped people in them.

Another scenario is when a wildfire occurs for example the fires that happened the summer of 2021 in Attica and destroyed 16% of the forests [12]. Along with the forest a lot of buildings suffered various damages to their structural integrity. Something similar happened to Evia Island due to the summer heatwave and other factors that are not my study field but left us with over 500.000 acres of burnt forest and of course a lot of damaged and even totally collapsed buildings. So, people living in these areas need compensation from the current government. Therefore, all the damaged buildings need to pass a visual inspection from government assigned civil engineers to create reports for each individual building. But when we are dealing with thousands of buildings across a wide area of an island like Evia something like this is very time consuming. To put it in a simpler way people, need to re-build/fix their homes in expectation of having a place to live and government must classify the damages quickly. Being able to use satellite images of the affected acres to generate a report with the help of deep learning model or even use

Epoch	main/loss	validation/main/loss	main/accuracy	validation/main/accuracy	elapsed time
01	0.248371	0.228309	0.918139	0.959325	72.3883
10	0.149629	0.0970564	0.939556	0.956784	600.198
20	0.117099	0.088584	0.953469	0.965297	1184.31
30	0.116498	0.0813691	0.954175	0.96966	1770.02
40	0.0957175	0.0730834	0.962931	0.97179	2359.9
50	0.0785189	0.0811975	0.969644	0.970451	2949.08
60	0.0831312	0.0682702	0.967713	0.974896	3552.44
70	0.0730851	0.0632934	0.971921	0.975701	4149.39
80	0.0747385	0.0515737	0.971226	0.979858	4740.75
90	0.0683723	0.0558712	0.974289	0.979437	5338.04
100	0.072806	0.0477328	0.972347	0.981581	5930.5

multiple UAVs to scan the whole affected area by divide and conquer method or a combination of both it will save a lot of time and money.

Performance

Using the SpaceNet algorithm to train our model so it can easily detect building so it can identify buildings in new locations that a new disaster might occur. We used santa-rosa-wildfire as a subset of our dataset to train our model to detect what is classified as a building. (Figure 4-5)

Figure 4 Accuracy of SpaceNet (Santa-Rosa-Wildfire)

Figure 5 Loss of SpaceNet (Santa-Rosa-Wildfire)

LinearSVM

For demonstration purposes if someone uses as a last layer a linear SVM he will get an accuracy of 25% of the classification based on the joint damage scale. This happens because the linear SVM will classify all buildings as one category. For example, if all building will be destroyed or will have a minor damage.

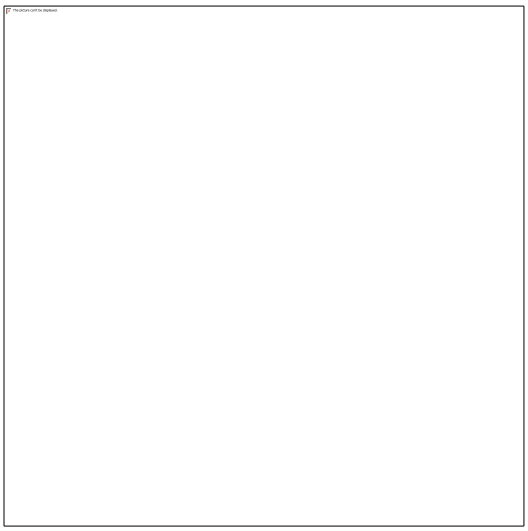


Figure 6 Damage Classification with Linear SVM (Santa-Rosa-Wildfire)

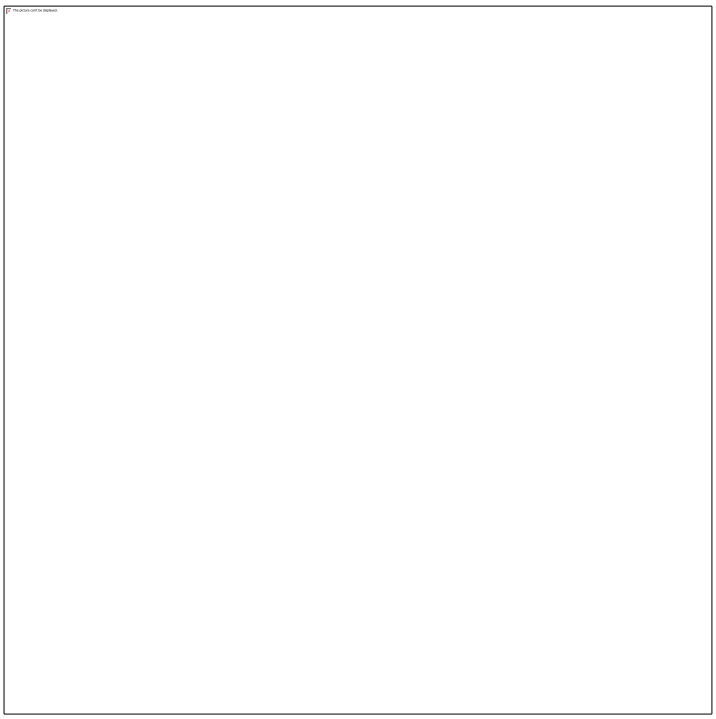


Figure 7 Damage Classification with Linear SVM (Santa-Rosa-Wildfire)

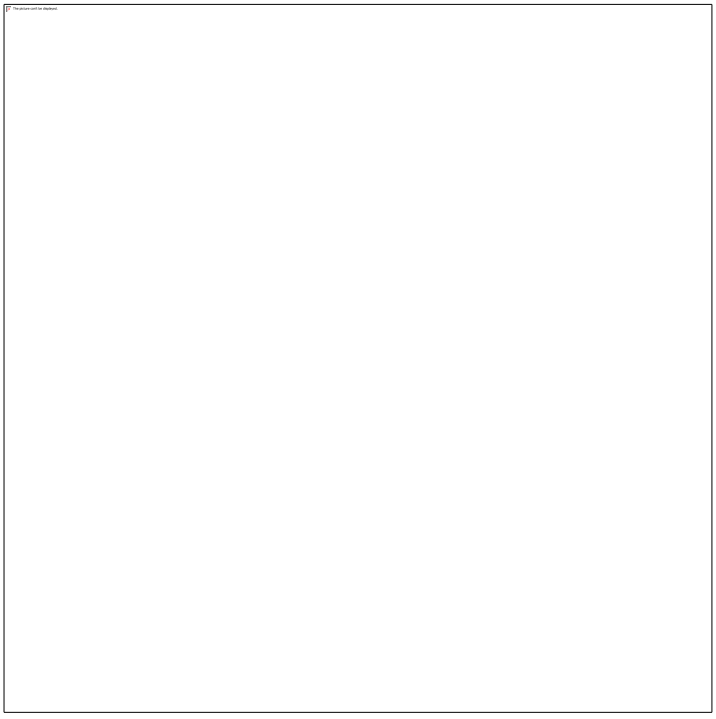


Figure 8 Damage Classification using baseline approach (Santa-Rosa-Wildfire)

Nonlinear-SVM

Running nonlinear SVM as the last layer of our damage classification model gives us reliable results but with some false positives but they are mostly misidentified buildings i.e. (top-right corner show two green “houses”) but that is caused from the building classification model that was only trained one hundred epoch and with a very small subset (Santa-rosa-wildfire) of the whole xView2 dataset.

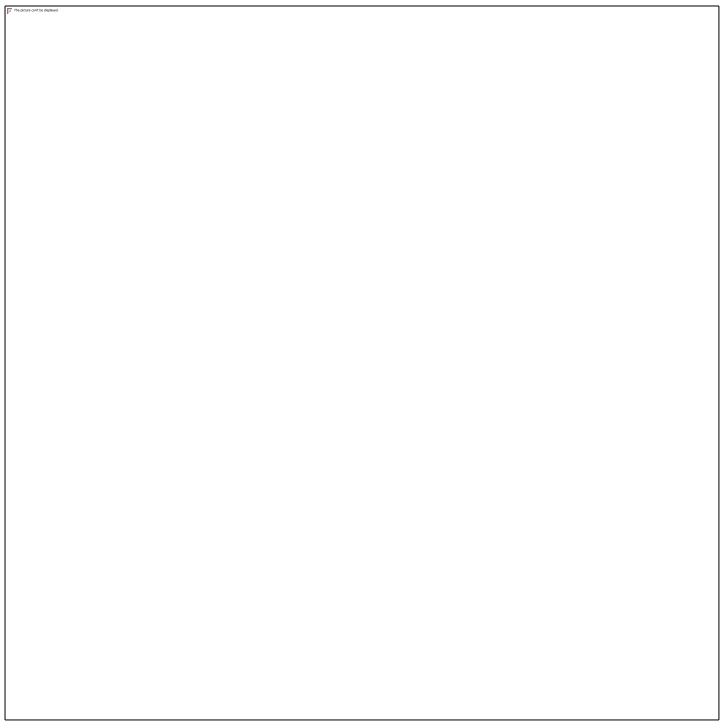


Figure 9 Damage Classification using Non-Linear SVM as last layer (Santa-Rosa-Wildfire)

Conclusion and Future Work

Taking everything into account, building detection can be a very challenging due to the fact that when you create or extend your dataset with images you have to deal with photos that have some noise (i.e., clouds covering part or the whole building, foggy/muddy area) but xview2 seem to merged a lot of useful data into one properly label dataset to help with a good baseline that can be extended relatively easier. In addition, damage assessment can be a very interesting research field as it can help in a lot of difficult situations i.e.

(earthquake, tsunami or even a war).So to contribute in this area we wanted to see if an SVM as it a very common last layer along with the soft-max approach in order to solve classification problems [19] and find out if it will benefit our damage classification model.

Future Work

To further improve the proposed model someone can use ArcGIS Pro to fetch building feature map of the affected area so the model will only perform damage classification because it will already know where the buildings are based on the extracted feature map. Another approach is to use Google Maps API to fetch satellite images of affected area days prior to the natural disaster tailored to his needs (e.g., same weather conditions as the disaster ones) and use SpaceNet or Building Footprint Extraction [13] to extract building feature map. The second one works well with both satellite images and aerial ones, so the use of a UAV becomes a lot easier. The only downside of this is that after the address of the natural disaster the model needs to be trained again with the new provided data.

Considering the above, the use of an SBC (e.g., Raspberry Pi) equipped with a camera on top of a drone with the help of a 5G capable model can easily send those images in real time (1ms) to a cloud infrastructure to perform the whole classification process and provide the drone of better the human that controls the drone a separate feed of video containing colored buildings corresponding to their damage.

Another approach of the propose of a drone is the use of the Jetson Nano to perform the AI workload on the device as Jetson is capable of such GPU heavy tasks without the need of cloud and save time for transmission to the cloud infrastructure and vice versa.

To further complete the entire process the whole dataset can be used to train the building classification algorithm and later train the damage classification per disaster. So, in any future disaster the specific model can be used to address that extremely specific disaster-problem and therefore have one model for every disaster that might occur in the future.

Bibliography

1. Xu, J. Z., Lu, W., Li, Z., Khaitan, P., & Zaytseva, V. (2019, October). Building Damage Detection in Satellite Imagery Using Convolutional Neural Networks. *arXiv:1910.06444 [cs, eess, stat]*. Retrieved October 27, 2021, from http://arxiv.org/abs/1910.06444

2. Gupta, R., Hosfelt, R., Sajeev, S., Patel, N., Goodman, B., Doshi, J., . . . Gaston, M. (2019, November). xBD: A Dataset for Assessing Building Damage from Satellite Imagery. *arXiv:1911.09296 [cs]*. Retrieved October 27, 2021, from http://arxiv.org/abs/1911.09296

3. He, K., Zhang, X., Ren, S., & Sun, J. (2015, December). Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*. Retrieved October 27, 2021, from http://arxiv.org/abs/1512.03385

4. Definition of SVM according to OpenCV Documentation https://docs.opencv.org/4.5.3/d1/d73/tutorial_introduction_to_svm.html

5. Nia, K. R., & Mori, G. (2017, May). Building Damage Assessment Using Deep Learning and Ground-Level Image Data. *2017 14th Conference on Computer and Robot Vision (CRV)* (pp. 95–102). Edmonton: IEEE. doi:10.1109/CRV.2017.54

6. Ronneberger, O., Fischer, P., & Brox, T. (2015, May). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597 [cs]*. Retrieved October 27, 2021, from http://arxiv.org/abs/1505.04597

7. Zhang, Z., Liu, Q., & Wang, Y. (2018, May). Road Extraction by Deep Residual U-Net. *IEEE Geoscience and Remote Sensing Letters, 15*, 749–753. doi:10.1109/LGRS.2018.2802944

8. Moradi, M., & Shah-Hosseini, R. (2020). Earthquake Damage Assessment Based on Deep Learning Method Using VHR Images. *Environmental Sciences Proceedings, 5*, 16. doi:10.3390/IECG2020-08545

9. SpaceNet (https://github.com/motokimura/spacenet_building_detection/)

10. Bengio, Y., Simard, P., & Frasconi, P. (1994, March). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks, 5*, 157–166. doi:10.1109/72.279181

11. Morales-Valdez, J., Alvarez-Icaza, L., & Escobar, J. A. (2020, July). Damage Localization in a Building Structure during Seismic Excitation. *Shock and Vibration, 2020*, 1–17. doi:10.1155/2020/8859527

12. Meteo.gr https://www.meteo.gr/article_view.cfm?entryID=1910

13. Building Footprint Extraction https://www.arcgis.com/home/item.html?id=a6857359a1cd44839781a4f113cd5934

14. NVIDIA Jetson: The AI platform for edge computing https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/

15. Strong open-source chess engine https://stockfishchess.org/

16. Machine learning, explained https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

17. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., . . . Chen, T. (2017, October). Recent Advances in Convolutional Neural Networks. *arXiv:1512.07108 [cs]*. Retrieved March 13, 2022, from http://arxiv.org/abs/1512.07108

18. Cortes, C., & Vapnik, V. (1995, September). Support-vector networks. *Machine Learning, 20*, 273–297. doi:10.1007/BF00994018

19. Kavzoglu, T., & Colkesen, I. (2009, October). A kernel functions analysis for support vector machines for land cover classification. *International Journal of Applied Earth Observation and Geoinformation, 11*, 352–359. doi:10.1016/j.jag.2009.06.002

20. Maulik, U., & Chakraborty, D. (2017, March). Remote Sensing Image Classification: A survey of support-vector-machine-based advanced techniques. *IEEE Geoscience and Remote Sensing Magazine, 5*, 33–52. doi:10.1109/MGRS.2016.2641240

21. Mountrakis, G., Im, J., & Ogole, C. (2011, May). Support vector machines in remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing, 66*, 247–259. doi:10.1016/j.isprsjprs.2010.11.001

22. Mantero, P., Moser, G., & Serpico, S. B. (2005, March). Partially Supervised classification of remote sensing images through SVM-based probability density estimation. *IEEE Transactions on Geoscience and Remote Sensing, 43*, 559–570. doi:10.1109/TGRS.2004.842022

23. Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92* (pp. 144–152). Pittsburgh, Pennsylvania, United States: ACM Press. doi:10.1145/130385.130401

24. Zhu, G., & Blumberg, D. G. (2002, May). Classification using ASTER data and SVM algorithms;. *Remote Sensing of Environment, 80*, 233–240. doi:10.1016/S0034-4257(01)00305-4

Section A

GPU: 0

Minibatch-size: 16

Crop-size: 400

Epoch: 100

epoch	main/loss	validation/main/loss	main/accuracy	validation/main/accuracy	elapsed time
1	0.248371	0.228309	0.918139	0.959325	72.3883
2	0.171758	0.184291	0.941087	0.959417	130.107
3	0.174309	0.165682	0.937529	0.957574	187.6
4	0.156248	0.130873	0.942095	0.958216	244.938
5	0.150401	0.155715	0.945608	0.941414	307.313
6	0.160608	0.110339	0.942958	0.956322	364.715
7	0.162063	0.105588	0.939092	0.959988	422.048
8	0.148145	0.108511	0.94729	0.954119	479.314
9	0.150408	0.100966	0.94088	0.958286	542.456

total [####.....] 9.76%

this epoch [##.....] 75.61%

100 iter, 9 epoch / 100 epochs

inf iters/sec. Estimated time to finish: 0:00:00.

10	0.149629	0.0970564	0.939556	0.956784	600.198
11	0.137875	0.0994239	0.945984	0.958634	657.487
12	0.143337	0.100507	0.94334	0.95379	714.853
13	0.142189	0.212782	0.94318	0.95973	777.116
14	0.137869	0.0964185	0.944517	0.959419	834.402

15	0.134803	0.0881785	0.945146	0.963328	891.748
16	0.13055	0.110902	0.948457	0.963406	949.162
17	0.141082	0.0873135	0.945165	0.964491	1011.66
18	0.142233	0.0955827	0.944351	0.966174	1069.22
19	0.119621	0.0849503	0.953885	0.967461	1127.02

total [#########.....] 19.51%

this epoch [############################.....] 51.22%

200 iter, 19 epoch / 100 epochs

0.17415 iters/sec. Estimated time to finish: 1:18:57.398682.

20	0.117099	0.088584	0.953469	0.965297	1184.31
21	0.121465	0.0816433	0.952036	0.965443	1246.43
22	0.126179	0.0872653	0.951859	0.966938	1303.42
23	0.107696	0.0772394	0.957968	0.968948	1360.53
24	0.112647	0.077409	0.956857	0.968368	1417.5
25	0.111414	0.0841082	0.957264	0.967176	1479.61
26	0.115445	0.092549	0.955892	0.968219	1536.62
27	0.127015	0.0863309	0.951334	0.964179	1593.64
28	0.111527	0.0822129	0.958583	0.967748	1650.82
29	0.10085	0.0808828	0.96147	0.965174	1712.68

total [################.....] 29.27%

this epoch [##############.....] 26.83%

300 iter, 29 epoch / 100 epochs

0.17466 iters/sec. Estimated time to finish: 1:09:10.910317.

30	0.116498	0.0813691	0.954175	0.96966	1770.02
31	0.104715	0.0898064	0.959501	0.962393	1832.73
32	0.111302	0.0715475	0.955923	0.972555	1890.05
33	0.124029	0.0994614	0.950924	0.96448	1952.59
34	0.115141	0.0824721	0.95641	0.966538	2010.29
35	0.127418	0.102276	0.95061	0.965954	2068.23
36	0.11104	0.0688331	0.956823	0.974216	2125.61
37	0.0997533	0.0797492	0.960572	0.969716	2187.98

38 0.108509 0.0764989 0.957282 0.96903 2245.14

39 0.101501 0.0716731 0.961042 0.972733 2302.46

total [####################.....] 39.02%

this epoch [#.....] 2.44%

400 iter, 39 epoch / 100 epochs

0.17357 iters/sec. Estimated time to finish: 1:00:00.922183.

40 0.0957175 0.0730834 0.962931 0.97179 2359.95

41 0.0889849 0.0697198 0.965659 0.9725 2422.83

42 0.109669 0.121636 0.955574 0.962107 2480.33

43 0.103904 0.0728264 0.959595 0.97026 2537.58

44 0.0952033 0.0896886 0.962459 0.963557 2594.95

45 0.0961043 0.0647265 0.962201 0.975304 2657.06

46 0.0898943 0.0804596 0.964735 0.97182 2714.16

47 0.091459 0.0717713 0.963961 0.970845 2771.52

48 0.103929 0.0699394 0.95989 0.971433 2829.12

total [#########################.....] 48.78%

this epoch [##.....] 78.05%

500 iter, 48 epoch / 100 epochs

0.17456 iters/sec. Estimated time to finish: 0:50:07.620641.

49 0.0828965 0.0705439 0.968148 0.971807 2891.61

50 0.0785189 0.0811975 0.969644 0.970451 2949.08

51 0.0914359 0.0633786 0.964509 0.97742 3021

52 0.0841678 0.0630484 0.967562 0.97641 3079.17

53 0.0947962 0.0857803 0.963663 0.967 3142.42

54 0.0814097 0.11928 0.968978 0.965359 3200.08

55 0.0940421 0.0779933 0.962563 0.972216 3257.88

56 0.0886972 0.0655407 0.964402 0.974286 3316.03

57 0.0859889 0.0648892 0.967084 0.973419 3379.1

58 0.0869022 0.068124 0.965733 0.975923 3436.88

total [##############################.....] 58.54%

this epoch [############################.....] 53.66%

600 iter, 58 epoch / 100 epochs

0.17334 iters/sec. Estimated time to finish: 0:40:51.759806.

59	0.0822741	0.0625559	0.968018	0.976533	3494.69
60	0.0831312	0.0682702	0.967713	0.974896	3552.44
61	0.0837534	0.0605224	0.966955	0.975377	3615.23
62	0.080858	0.0950384	0.969182	0.970821	3673.19
63	0.0858145	0.0697039	0.967085	0.974962	3730.97
64	0.0804658	0.0634525	0.968379	0.973715	3789.33
65	0.0820634	0.0679786	0.96692	0.976543	3853.05
66	0.0747686	0.0624412	0.970501	0.976478	3911.42
67	0.0778439	0.0587748	0.969718	0.977483	3969.79
68	0.0750105	0.0651996	0.971134	0.976345	4027.87

total [###.....] 68.29%

this epoch [################.....] 29.27%

700 iter, 68 epoch / 100 epochs

0.17314 iters/sec. Estimated time to finish: 0:31:17.112641.

69	0.0731202	0.0554982	0.97163	0.979046	4091.22
70	0.0730851	0.0632934	0.971921	0.975701	4149.39
71	0.0725655	0.0691743	0.971785	0.974449	4207.55
72	0.0759979	0.0567736	0.970344	0.979041	4265.78
73	0.0716494	0.0862464	0.97156	0.972297	4328.97
74	0.0765689	0.0569171	0.97015	0.97768	4387.21
75	0.0651007	0.0570131	0.97516	0.978539	4445.25
76	0.0677926	0.0536317	0.973084	0.979542	4503.42
77	0.0757188	0.0595677	0.970207	0.976759	4566.5
78	0.0674326	0.0638738	0.97378	0.977465	4624.64

total [##.....] 78.05%

this epoch [##.....] 4.88%

800 iter, 78 epoch / 100 epochs

0.17282 iters/sec. Estimated time to finish: 0:21:41.967752.

79	0.0689469	0.0622197	0.973102	0.977563	4682.76

80	0.0747385	0.0515737	0.971226	0.979858	4740.75
81	0.0727194	0.0550089	0.971974	0.980278	4804.09
82	0.0643093	0.0571189	0.975393	0.977822	4862.13
83	0.0767177	0.0588132	0.969083	0.977863	4920.19
84	0.0722139	0.0571848	0.971718	0.977924	4978.26
85	0.0679254	0.0552411	0.973352	0.979098	5041.69
86	0.0706984	0.0503689	0.972849	0.981077	5099.83
87	0.0628854	0.0488895	0.975805	0.980857	5158.02

total [##.....] 87.80%

this epoch [##.....] 80.49%

900 iter, 87 epoch / 100 epochs

0.17313 iters/sec. Estimated time to finish: 0:12:02.013041.

88	0.0696999	0.0541805	0.973247	0.97934	5216.16
89	0.0657108	0.0556194	0.974176	0.978367	5279.57
90	0.0683723	0.0558712	0.974289	0.979437	5338.04
91	0.0675669	0.0502057	0.974191	0.980271	5396.36
92	0.0740278	0.0538187	0.97133	0.979509	5454.57
93	0.064117	0.0519578	0.975001	0.980471	5517.75
94	0.0628196	0.0479857	0.976026	0.981419	5576.17
95	0.0604761	0.0703855	0.976585	0.973775	5634.91
96	0.0700984	0.0497585	0.9724	0.981013	5693.68
97	0.0777764	0.0600608	0.970134	0.978627	5756.87

total [###...] 97.56%

this epoch [################################.....] 56.10%

1000 iter, 97 epoch / 100 epochs

0.17293 iters/sec. Estimated time to finish: 0:02:24.567267.

98	0.0752757	0.0687284	0.971306	0.975276	5814.82
99	0.071402	0.0537116	0.971968	0.979474	5872.67
100	0.072806	0.0477328	0.972347	0.981581	5930.5