Satellite Characterization via Restoration-Free Imaging: A Novel Machine Learning Paradigm for SSA

Ryan Walden
Georgia State University, Atlanta, GA
rwalden3@student.gsu.edu

Taslim Dosunmu
Georgia State University, Atlanta, GA
tdosunmu@student.gsu.edu

Stuart M. Jefferies
Georgia State University, Atlanta, GA
Institute for Astronomy, University of Hawaii, HI
sjefferies@gsu.edu

Daniel Pimentel-Alarcón
University of Wisconsin-Madison, Madison, WI
pimentelalar@wisc.edu

ABSTRACT

In this paper, we investigate several state-of-the-art deep convolutional neural network architectures using a lightweight Bayesian neural architecture search strategy to classify images of satellites affected by varying levels of simulated atmospheric distortion. This paper provides a comprehensive study of the state-of-the-art for deep learning in SSA. Over the course of 392 trained models, we obtain a final, repeatable model with over 90% accuracy on unseen images with varying amounts of atmospheric distortion. Our approach demonstrates that sufficiently deep convolutional neural networks implicitly learn to ignore or remove atmospheric distortion when tasked with classification alone, mitigating the necessity for image restoration techniques for satellite characterization.

1. INTRODUCTION

For the past few years, space domain awareness has hinged on high-resolution restoration of satellite imagery that has been severely distorted by the Earth's turbulent atmosphere. Restoration of this sort is a computationally demanding procedure that typically requires screening and processing thousands of images; this can take hours of compute time on dedicated servers, impeding real-time analysis. Restoration, however, is only an intermediate step towards more succinct goals, such as identifying the satellite model, orientation, health, and intent. To the human eye, image restoration is an unavoidable prior to extracting these satellite characteristics, as our brains are not trained to distinguish between distorted images directly. However, the fact that restoration is even possible suggests that all the necessary information to recover the desired characteristics is indeed embedded in the distorted images, only encoded in a different way due to the light diffraction patterns produced by the atmosphere.

Hence, we hypothesize that by training a deep convolutional neural network (D-CNN) to interpret distorted images directly, we can recover the desired satellite characteristics (e.g., model and orientation) in one step, with no need for image restoration nor further processing. Our rationale is that convolutional filters are qualified by design to undertake

All results from experiments are publically available at https://app.wandb.ai/rosenblatt/satellite-model-and-orientation/sweeps

convolution transformations, like the one used to model the image formation process and estimated by prior works (see Section 2). To determine an ideal D-CNN architecture and configuration, we utilize a lightweight Bayesian neural architecture search (NAS) strategy to converge on a final model from a selection of several architectures, many potential hyperparameter configurations, and multiple training and initialization strategies. Training a network of this sort requires a sufficiently large dataset, rich enough to capture all the spectrum of satellites' variability. To this end we introduce a novel synthetic data generation strategy that combines a state-of-the-art 3D graphics engine with an atmospheric distortion pipeline to generate realistic samples of satellite models at various orientations and distances from an observatory.

2. PRIOR WORK

2.1 Model Approaches

Customarily, researchers have incrementally improved various methods to better estimate and reduce the atmospheric distortion of an image [6], [8]. These methods are convolutional by design and often require 1000s of continuous frames of a single object to produce a quality reconstruction. Furthermore, these methods converge on their results by minimizing an error function which serves as a conduit for the noise/distortion in the image. A common approach to utilizing convolutions in deep learning to estimate a noise function is by integrating them as neurons inside a deep neural network; referred to as a D-CNN. In references [9] and [10], D-CNNs were investigated for astronomical image denoising and noise estimation respectively. Both of these works show competitive results to their predecessors, but like their predecessors, neither produce any characterization predictions or estimates of the subject of the image – the satellite. Instead, researchers have relied on human specialists or further supervised methods to produce characterization results.

Some attempts have been made to automate this process altogether. Reference [11] used a neural network to investigate the end to end method described but have many clear distinctions from our approach. In their future works, they directly commented on the potential of CNNs, but only performed experiments with a traditional dense neural networks in their paper. Furthermore, in our paper we report confidence metrics for our best model and accuracy metrics for all experiments in addition to investigating architectures several orders of magnitude larger. Neural architecture search (NAS) has also been investigated in SSA, but our work is the first to utilize it for satellite imagery classification [18]. Our paper comprehensively covers both state-of-the-art D-CNNs and NAS and arrives on the conclusion that reconstruction-to-characterization methods can be replaced by an end to end D-CNN model, achieving the desired results in a single step.

2.2 Dataset Approaches

In previous works, researchers have used several methods to build satellite image datasets: use a pre-existing dataset such as SatSim or SatNet [5], capture the images using telescopes and other equipment, use a rendering tool such as QUID to create synthetic images [11], or use a hybrid approach where satellite images are captured using telescopes, but digitally enhanced to meet their requirements [10].

We were initially provided a Matlab program capable of rendering 3D objects and applying approximate atmospheric distortion filters [6], but we found it inflexible as it is difficult to set up a licensed copy in our cloud working environment; further, the language syntax is not as common as other languages', making modifications challenging to implement. We instead opted to use Python, which is portable to nearly all systems, to perform image distortion, and we also used Unity to handle rendering the base satellite object images– though, any program that can render 3D objects will work in its place.

3. SYNTHETIC DATASET

Adequately training a neural net requires a large amount of training data. We were not able to find a repository with enough images of satellites that satisfied our requirements, so we developed our own synthetic data generator. The requirements for the satellite images that we use to train the model are: 1. the images must be representative of what would be captured by real equipment, 2. there must be a large number of diverse images for each type of satellite, and 3. there must be a way to categorize characteristics of the satellite images (e.g. distortion level, type of satellite, and orientation of the satellite). We built the synthetic dataset generator pipeline using Unity for generating non-distorted

images of satellite objects, and a Python script that applies a representative approximation of atmospheric distortion using a Kolmogorov phase screen to generate the point spread function which is then convolved with the original satellite image. With this pipeline, we are able to generate thousands of satellite images within minutes on a standard personal computer with parameterized configurations for the previously mentioned characteristics.

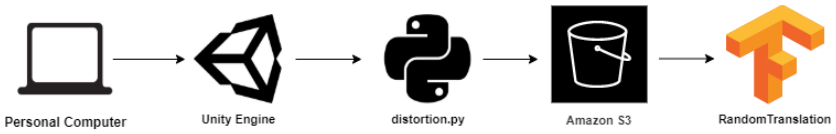

Fig. 1: A high-level overview of the service orchestration for generating a synthetic dataset

3.1 Pipeline Details

We designed the synthetic data generator pipeline to be flexible and easily configurable. A user is only required to provide 3D mesh object files of satellites of interest to the Unity object imaging component. We used 18 satellite object files that are provided by NASA's open source 3D model library [12]. The Unity component generates the desired number of images of a satellite rotated in three axes by degree values randomly generated within a uniform distribution. Because the language used to program Unity Engine scripts is C#– a C-like language with commonly used syntax– it is trivial to change its behavior [17]. The generated images are then processed by the distortion script that can take the following configuration arguments: a telescope aperture size (meters), the Fried parameter (meters), the outer scale (integer), a stencil length factor (integer), and an optional seed value to control randomness. These arguments control the Kolmogorov phase screen that is used to replicate atmospheric distortion; particularly the aperture size, and Fried parameter have a significant impact on the resultant phase screen filter. How heavily distorted the image is can be expressed using the following equation:

$$distortion\ level = \frac{aperture\ size}{Fried\ parameter} \tag{1}$$

A distortion level of 5 is low distortion, 14 is medium distortion, and >21 is high distortion.

Fig. 2: Images of a Chandra satellite. Distortion levels from left to right: 0, 5, and 21.

3.2 Datasets Used for Model Development and Evaluation

Creating datasets with characteristics conducive to learning is critical to training and evaluating a high accuracy model that does not overfit. Good datasets have a large number of samples, ideally even distribution between satellite classes, and representation of the expected images that would be seen at inference time. To build our datasets, we generated an equal number of images for each satellite class, and mixed images with no distortion, and images with low and high distortion (distortion levels of 5 and 21 respectively) with a split of 15% no distortion, 75% low distortion and 10% high distortion. Further, our training and validation split is 80% and 20% respectively.

Determining how many images are necessary for training is more of an art than a science; 1,000 images per class is a common choice that sees successful results in computer vision model training [4]. We initially included only 1,000

images in our datasets, but found that our models were underfitting (see Section 6.2). We then increased the image count to 5,000 and consistently saw significantly greater performance (see Section 6.3). This is reasonable because, with a uniform distribution of random orientations, there is a direct proportional relationship between the number of images, the granularity of orientations covered by the dataset, and the amount of information that a model can learn about the satellite objects. Thus, the finer the granularity, the more knowledge that can be learned from the dataset.

Along with the 1,000 and 5,000 image-per-class datasets we created, we also created three variations of a structured dataset for model evaluation that, rather than being composed of random orientations, is composed of satellite objects that are rotated by a specific step-size until 360° is covered in each axis. The step-size we used was 30°, resulting in 1,728 images per satellite class $[(360^\circ/step\_size)^3]$. The three variations of this structured dataset are the same set of images without distortion, with a distortion level of 5, and a distortion level of 21.

Referring back to the choice of the number of images per class, if an angular granularity of 30° results in 1,728 images, then a granularity of 36° results in 1,000 images, and a granularity of 21° results in 5,000 images, reinforcing our claim about the relationship between image count and information that can be learned.

3.3 Random Translation Layer

During training and validation steps, we apply data augmentation to images in our dataset using a RandomTranslation layer inserted before the input layer of our model. For all experiments, we utilized a variance factor of 40% for both width and height with a 0 pixel intensity (black) fill. When evaluating our best model in Section 7.2, the translation layer was removed due to environment constraints. This caused very little change in overall accuracy with a net increase of only 0.3%, thus we believe this does not hinder our ability to accurately evaluate and interpret the model.

4. NEURAL ARCHITECTURE SEARCH

Neural architecture search (NAS) is a modern deep learning technique which automates the search for a best model given a set of architecture and hyperparameter constraints. This section describes the rationale for the constraints we set for this search, the optimization strategy, and the underlying optimization algorithm.

4.1 Bayesian Optimization

We used a Bayesian search optimizer to guide our neural architecture search. Bayesian search differs from traditional search strategies (grid and random) due to its ability to build a probability model of the objective function and use it to select the most promising parameters. In our case, the objective function is defined as the resulting model's validation classification accuracy given the parameters used to configure the architecture. This results in significantly less experiments to converge on performant models than traditional search strategies. Furthermore, this means a larger search space is feasible, motivating our decision to allow loss weights to be bound on a continuous range (see Figure 4.b). The sections below describe the various parameters that were configured by the Bayesian optimizer.

4.2 Architectures Investigated

In more advanced NAS, the architecture for a given task is developed "from scratch" by the search algorithm [7]. We decided this was out of scope for this paper and found it more sensible to first survey existing state-of-the-art computer vision architectures. We focused primarily on models that utilize residual connections, which have been proven to outperform their vanilla counterparts in a variety of computer vision tasks [15]. For our search, we selected 4 base architectures: VGG-16, ResNet50v2, DenseNet201, and EfficientNet-B4. Of these architectures VGG-16 was the only one lacking residual connections and primarily served as a counter-example to test our hypothesis that residual models would perform better on this task. The remaining architectures all utilized variations of residual connections. Further, the residual model architectures have a similar number of trainable parameters (see Figure 3), to provide a fair comparison.

4.3 Transfer Learning

In transfer learning, models are initialized using pre-trained weights from a previous task. The rationale is that the features learned from this task may already be applicable to the new task. In this case, only some layers of the model need to be trained to converge on performant results. Since all of the base architectures surveyed were available in TensorFlow's Keras API [3], their pre-trained ILSVRC ImageNet dataset weights were readily available as well. Unfortunately, our dataset does not qualitatively reflect the images in the ImageNet dataset). To our benefit, we

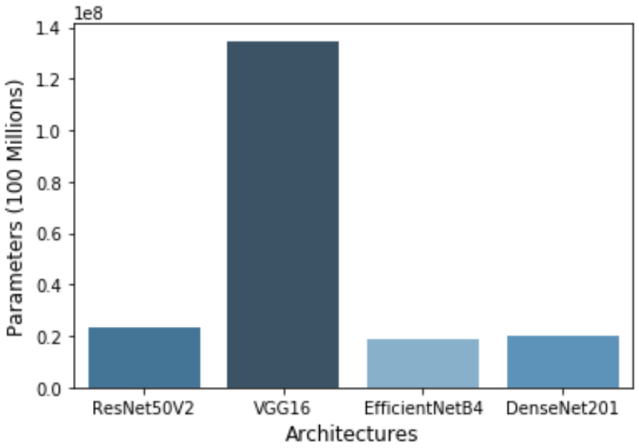

Fig. 3: A bar plot of the number of parameters in each base architecture.

explicitly control the number of images in our dataset and prior works have suggested that in such a scenario, we are still likely to benefit from using a fine tuning strategy. In this case it is recommended to initialize the model with pre-trained weights, and allow all weights of the model to be trained [16]. Since we are retraining all of the layers, we decided to use transfer learning as a NAS parameter via a simple on/off configuration (denoted by the "imagenet" values in Figure 4.c).

4.4 Multi-Task Learning

In multi-task learning, models perform multiple predictions. This can be implemented in a wide variety of ways. For our approach, we wanted to investigate if joint training on orientation data could lead to faster or more performant convergence on our primary task, satellite model classification. In joint training, both tasks produce individual loss values which are summed and then optimized using the same function (see Figure 4.a). In our strategy, each task receives a dense layer which will only forward-propagate to its respective output layer. This strategy is inherently problematic if the magnitude of error is significantly larger for one task than another. To address this, we allow for loss weights to be optimized by the NAS optimizer. Since this optimizer maximizes classification validation accuracy (our primary task), we rationalize that if orientation is not beneficial to the primary task it will simply reduce the loss weight of the orientation task towards zero. To avoid this useless optimization we additionally allow for a simple on/off configuration (denoted by the "orientation" values in Figure 4.c). Additionally, orientation data is cyclical. To account for this, we perform cyclical feature engineering which involves applying the appropriate trigonometric functions to the original degree values. This comes with the benefit of preserving the cyclical relationship such that the encoding for 1 degree is close to the encoding for 360 degrees as well as normalizing our output between 1 and 0. For three axes of rotation, this results in an output layer of 6 neurons.

4.5 Model Hyperparameters

To converge on a best model, it is imperative to optimize hyperparameters. Since all of our models' base architectures are predefined, we only optimized the parameters of the additional layers. In Figure 4.c, these hyperparameters and their respective ranges are displayed. Some of these ranges are random, such that they only provide us with interpretable trends for model optimization as well as the importance of the parameter to our validation accuracy (see Figure 6). In addition, two of the hyperparameters, the activation function and optimizer, were fixed. Although experimenting with these parameters could lead to improved learning, both the Adam optimizer and Rectified Linear Unit (ReLU) activation function are the most widely used in deep learning.

4.6 Early Stopping

For the majority of neural architecture searches– both automated and by hand– the optimizer is tasked with minimizing validation loss of a model. Because we allowed for our optimizer to select loss weights when experimenting with a multi-task approach, minimizing loss was infeasible. Under our search scenario, the optimizer would always select a multi-task approach, and proceed to minimize the loss weights. To avoid this, we instead choose to optimize for validation classification accuracy, but this also raises its own unique problems. A network which maximizes validation

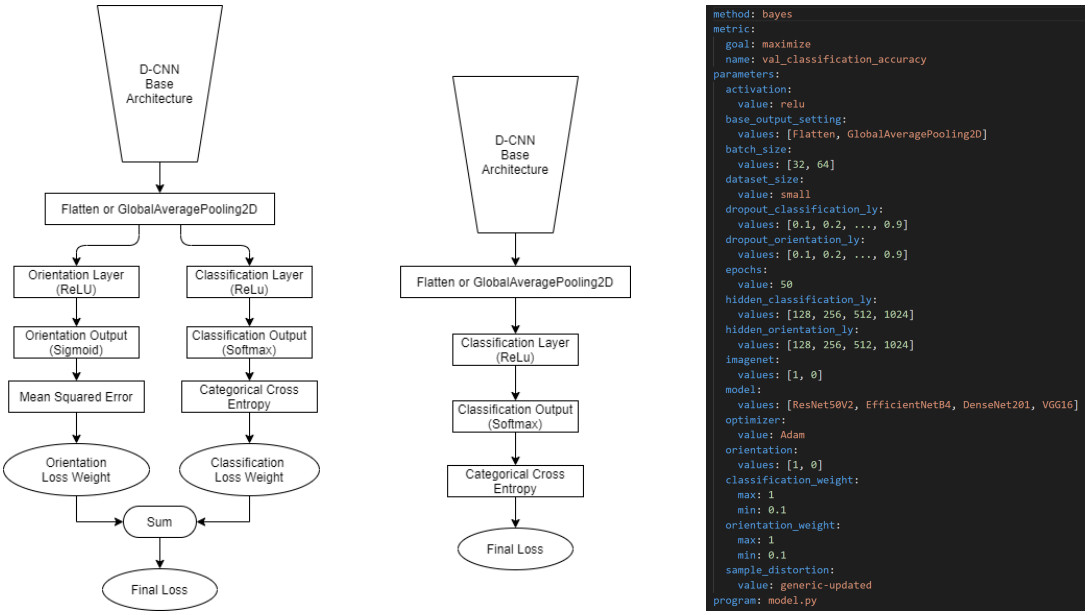

Fig. 4: (a) A multi-task architecture. (b) A vanilla approach to classification. (c) Neural architecture search parameters.

accuracy may converge on a high accuracy but low confidence model– the objective is to have both high accuracy and high confidence. To this effect, we utilize an early stopping strategy which halts model training if validation loss increases. Instead of optimizing a maximum epoch parameter, we simply set a large maximum epoch threshold (50) to prevent overtraining (denoted by ”epochs” in Figure 4.c).

5. CLOUD INFRASTRUCTURE

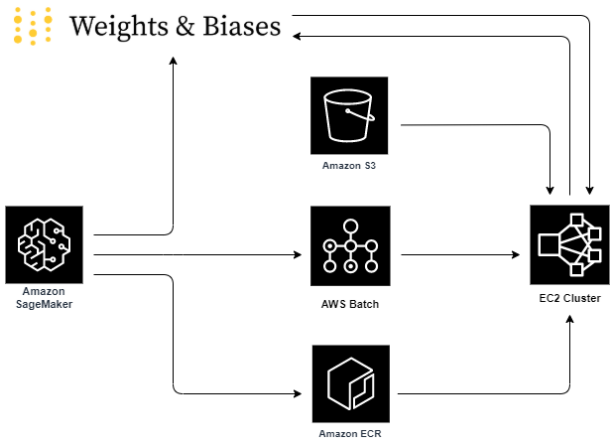

Fig. 5: A high-level overview of the service orchestration for performing nerual architecture search sessions.

5.1 Amazon Web Services (AWS)

AWS proved to be an essential tool to the scalability of our project, since we did not have on-premise access to any high compute GPUs necessary for comprehensive NAS [1]. Additionally, deep learning approaches are notorious for their issues with repeatability, especially in replicating the training environment. By using services available globally to AWS account members, we hope our strategy can be more easily repeated.

5.1.1 SageMaker & S3

Amazon SageMaker is AWS's flagship data science platform. It provides a JupyterLab environment to develop and test models and pipelines before launching exhaustive NAS. To be cost effective, the majority of our development was done using CPU instances. When tests involving model training were required, we switched the instance type to one with a GPU, giving us the flexibility we needed for different phases of model development. AWS Simple Storage Service (S3) was chosen as the most intuitive storage system for our dataset.

5.1.2 Elastic Container Registery (ECR) & Docker

Once model development was complete, it was imperative to create a version-controlled Docker image of the training process. Using Docker images provided high level repeatability such that any compute instance with Docker installed and sufficient memory could run our image. We used ECR to store different versions of our image which could then be accessed throughout our AWS environment for testing and experiments.

5.1.3 Batch, & Elastic Compute Cloud (EC2)

Although sufficiently large GPU instances were available on SageMaker, these instances were not cost optimized. AWS Batch allowed us to run training jobs on a Spot instance EC2 cluster, with flexible limitations on hourly rates. By starting our Batch jobs programmatically, we were able to quickly launch multiple training jobs asynchronously on individual nodes which reported their results to the same NAS Bayesian optimizer (see details in Section 5.2.1). During some experiments, we had as many as 16 independent instances for a significant fraction of the cost compared to traditional on-demand instances.

5.2 Weights and Biases

While AWS provides repeatability, Weights and Biases (Wandb) provides interpretability [2]. This service allowed for automated documentation of a variety of metrics describing a model. The automated documentation is hosted on a shareable dashboard and includes interactive graphs and aids in comparing runs under the same project. In recent years, major deep learning research organizations such as Toyota Research Institute and OpenAI have promoted and utilized this service in their papers as it is completely free for academics [14], [13]. To provide rich documentation of all of our experiments, we utilized Wandb for all of our training jobs which are publicly available at https://app.wandb.ai/rosenblatt/satellite-model-and-orientation/sweeps.

5.2.1 Sweeps

In addition to rich documentation, Wandb provides us with the Sweeps feature which manages our NAS strategy. When launching a sweep session, Wandb hosts a central controller and coordinates between the "agents" that execute training jobs. In our case, any instance we launch during a NAS session is considered an agent of that session. When a model has completed training, the agent requests the next model configuration from the central sweep server. This enables us to perform distributed deep learning, for which scaling is only constrained by cost and Wandb rate limits. Sweep sessions provide additional automated documentation for comparing configuration feature importance, and the convergence trend of experiments over time as they optimize for a particular metric– in our case, validation classification accuracy (see Section 4.4 and 4.6 for rationale).

6. EXPERIMENTS

6.1 Early Tests

In our initial tests, we focused on exploring the performance between residual CNNs and non-residual CNNs. In these tests we did not perform hyperparameter tuning, but did use transfer learning via fine tuning. We explored two architectures, ResNet50v1 and VGG-16 on classifying a dataset of only highly distorted images (D/r0=22). To our surprise, VGG-16 failed to converge on this task altogether, scoring a maximum validation accuracy of 6.48% whereas ResNet50v1 achieved over 90% accuracy in multiple experiments. This strengthened our hypothesis that

residual CNNs would likely be better suited for filtering information from atmospheric distortion, and motivated our selection of DenseNet201 and EfficientNet-B4 as additional model architectures to search.

6.2 Session 1 Part 1

The configuration for our initial NAS session is described in Figure 4c. Of all sessions, this was the shortest with only 61 complete runs with early stopping. The best model had only 38.01% validation accuracy. After reviewing the top models from this session, it was discovered that the models were experiencing catastrophic overfitting on their final epochs, losing as much as 29% validation accuracy in the case of our best model.

6.3 Session 1 Part 2

After observing overfitting and underfitting in the first session, we decided to increase the number of samples per class from 1,000 to 5,000. This second session was unique in that it was paused and resumed. A major benefit of the Wandb Sweeps feature is its ability to resume runs, maintaining the existing data used by the Bayesian optimizer. With exception to the increased dataset, this session utilized the same configuration as Session 1 Part 1 described in Figure x. During this session we realized that we had not performed cyclical feature engineering of the orientation data; we paused the session to correct this. At the time of pausing, the session had preformed 83 complete runs and proved significantly more successful, converging on a best model with a highest validation accuracy of 84.15%. After cyclical feature engineering was implemented for the orientation data, the session was resumed for another 92 runs, totaling 175 runs for the session. Overall, the best model for the session scored a validation accuracy of 86.06%, more than double the top score from the previous session.

6.4 Session 2

For the final session, we used our findings from the previous session to narrow down several of the configuration parameters. The details of our findings are available in section x and the configuration used for this sweep is displayed in Figure x. Because we had narrowed down several of the parameters, we decided to start a new session. In this final session, we focused entirely on optimizing the ResNet50v2 architecture, the best model architecture from previous sessions. After 155 runs, the session concluded with a best model achieving a highest validation classification accuracy of 90.46%.

7. RESULTS

In this section, we analyze the NAS results from Session 1 Part 2 and Session 2 and determine some general design principles towards an ideal architecture. Furthermore, we perform extensive analysis of our best model and consider what areas it can improve in.

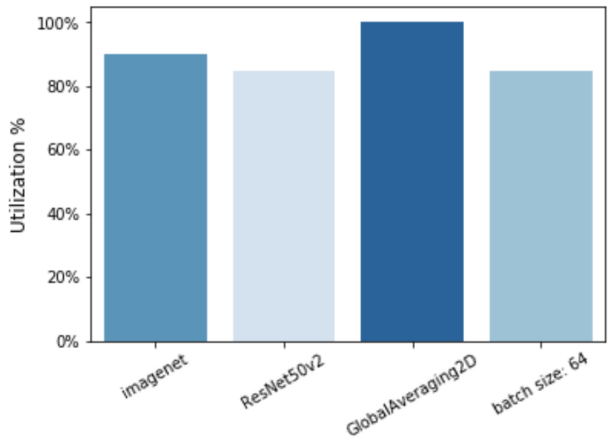

Fig. 6: Parameter usage percentage over the top 20 runs from Sweep Session 1 Part 2.

7.1 Neural Architecture Search Findings

The single largest performance increase in our experiments resulted directly from increasing the size of the dataset. Although we do not believe 5000 samples is particularly specific number, we now have an idea of the necessary rank to achieve over 90% accuracy given the architectures searched and the number of classes. Overall, we felt the NAS performed well in some areas, but there are still areas to explore. In sweep Session 1 Part 2, GlobalAveragePooling2D, transfer learning via ImageNet initialization , ResNet50v2, and larger batch sizes emerged as reliable features for an ideal architecture (see Figure 6). On the other hand, the effectiveness of multi-task learning proved inconclusive with only 20% utilization by the top 20 models of Session 1 Part 2, and 55% utilization of the top 20 models in Session 2. Notably, residual model architectures significantly outperformed VGG-16. Over the course of 19 runs, VGG-16 converged on a best validation classification accuracy of only 6.45%.

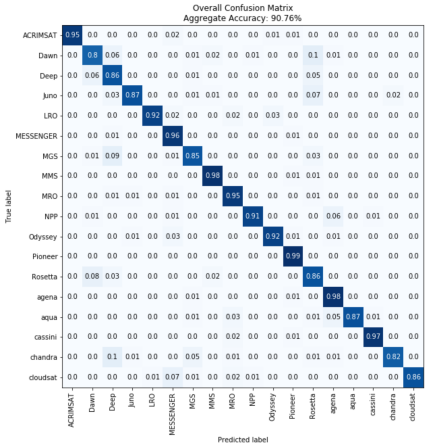

Fig. 7: Confusion matrix for the best model on the entire dataset.

7.2 Best Model Performance

Overall, our best model converged well on our dataset and showed limited confusion with classification (see Figure 7). Nonetheless, we were most interested in how our model preformed with respect to distortion level. In Figure 8, accuracy diminishes with increased distortion but at a non linear rate. Although we only trained on 3 levels of atmospheric distortion, we were interested how well the model generalized to distortion altogether and evaluated our best model on a dataset it had not seen before with distortion level D/r0=14 (see Figure 9). Our evaluation of this distortion level suggests that comprehensive training across all distortion levels is unnecessary– an important revelation towards an ideal dataset. In addition, it seems some objects in particular are much easier to discern than others, with 5 classes showing $> 90\%$ accuracy at the highest level of distortion.

8. FUTURE WORK

Although we have effectively demonstrated a successful approach towards restoration-free satellite characterization, there are many aspects of characterization not covered in our approach. Satellite component classification, satellite purpose characterization, image reconstruction, multi-sensor and video feed incorporation were not discussed or experimented with. Our goal is to continuously iterate on current research until we accomplish comprehensive satellite characterization. Some steps to achieving this include increasing our number of classes by at least one order of magnitude, investigating a combination of advanced neural architecture search and synthetic dataset optimization strategies, and maximizing hardware utilization for efficient training.

9. CONCLUSION

This work demonstrates great progress towards restoration-free satellite characterization. We believe further research in multi-tasking, dataset design, and neural architecture search of residual D-CNNs will lead to performance gains that

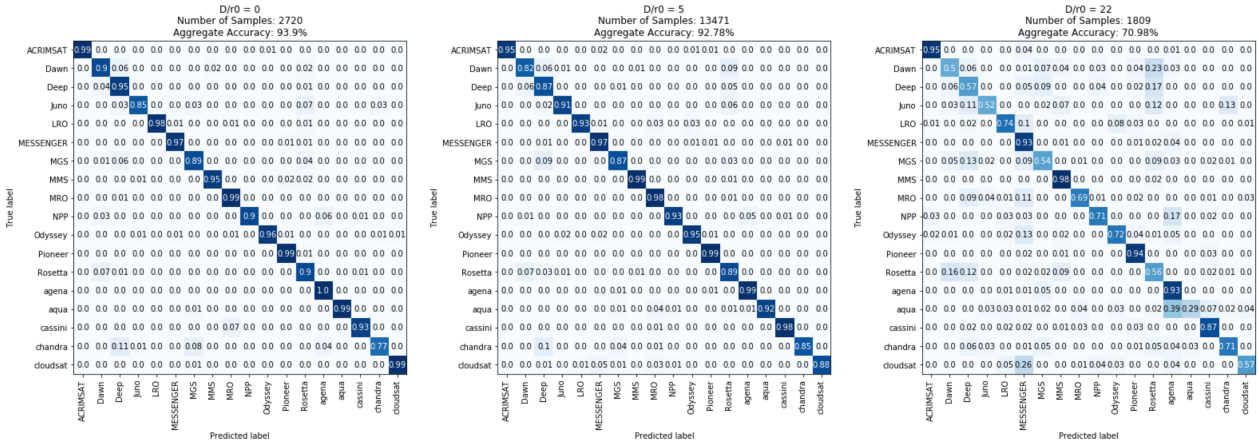

Fig. 8: Confusion matrices for the best model grouped by distortion level (D/r0).

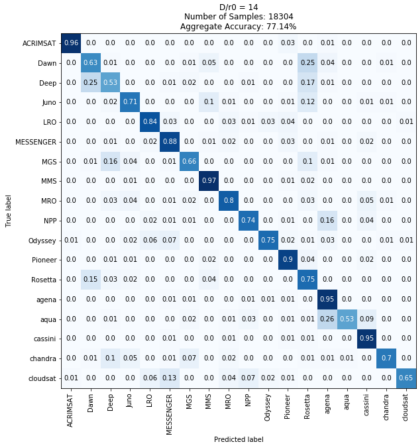

Fig. 9: Confusion matrices for the best model at distortion level D/r0=14.

will provide real-time satellite characterization. Additionally, our approach serves as a roadmap towards performing deep learning at scale in this field via neural architecture search, synthetic datasets, and cloud computing. Our team believes wide spread adoption of these technique is imminent as demands for space situational awareness grow from both the private and government space industry. As these techniques are adopted, it is important that we strive for explainable and repeatable research.

10. ACKNOWLEDGEMENTS

We thank Justin Hromalik (independent) for his assistance with AWS Batch and consultation; we thank Jack Le (independent) for his assistance with Unity Engine and advice on future works regarding synethic datasets; we thank Kehinde Adedara (Georgia State University) for his input, and advice throughout the research. We thank Professor Jonathan Shihao Ji (Georgia State University) for his consultation and review of initial research; we thank Professors Daniel Pimentel Alarcon and Stuart Jefferies for their direct advising over the entire length of our research. This work was personally supported by Ryan Walden.

11. REFERENCES

References

[1] Amazon. Aws: The world's most comprehensive and broadly adopted cloud platform, 2020.
[2] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.
[3] Google Brain. Tensorflow: An end-to-end open source machine learning platform, 2020.
[4] Jia Deng. Imagenet: A large-scale hierarchical image database, 2009.
[5] Justin Fletcher. Feature-based satellite detection using convolutionalneural networks, 2019.
[6] Stuart M. Jefferies. High-fidelity imaging using compact multi-frame blind deconvolution, 2017.
[7] Texas AM University DATA Lab. Autokeras, 2020. An AutoML system based on Keras.
[8] Jacob Lucas. Image enhancement for astronomical scenes, 2013.
[9] Jacob Lucas. Recovering astronomical images with deep neural network supported bispectrum processing, 2018.
[10] Jacob Lucas. Automated resolution scoring of ground-based leo observations using convolutional neural networks, 2019.
[11] Leon Muratov. Use of ai for satellite model determination from low resolution 2d images, 2019.
[12] NASA. Nasa 3d resources, 2020.
[13] Carey Phelps. Why experiment tracking is crucial to openai, 2018.
[14] Carey Phelps. An interview with tri, 2019.
[15] Microsoft Research. Deep residual learning for image recognition, 2015.
[16] Stanford. Transfer learning, 2020. CS231n Convolutional Neural Networks for Visual Recognition.
[17] Unity Technologies. Unity engine, 2020. A real-time 3D development platform.
[18] Dwight Temple. Network enabled - unresolved residual analysis and learning (neural), 2017.