Article

CNN Classification Architecture Study for Turbulent Free-Space and Attenuated Underwater Optical OAM Communications

Patrick L. Neary ^{1,*}, Abbie T. Watnik ^{2,†}, Kyle Peter Judd ^{2,†} and James R. Lindle ^{3,†}
and Nicholas S. Flann ^{1,†}

¹ Department of Computer Science, Utah State University, Logan, UT 84322, USA; nicholas.flann@usu.edu

² U.S. Naval Research Laboratory, Washington, DC 20375, USA; abbie.watnik@nrl.navy.mil (A.T.W.);
kyle.judd@nrl.navy.mil (K.P.J.)

³ DCS Corporation, Alexandria, VA 22310, USA; james.lindle.ctr@nrl.navy.mil

* Correspondence: patrick.neary@sdl.usu.edu

† These authors contributed equally to this work.

Received: 5 November 2020; Accepted: 1 December 2020; Published: 8 December 2020

Abstract: Turbulence and attenuation are signal degrading factors that can severely hinder free-space and underwater OAM optical pattern demultiplexing. A variety of state-of-the-art convolutional neural network architectures are explored to identify which, if any, provide optimal performance under these non-ideal environmental conditions. Hyperparameter searches are performed on the architectures to ensure that near-ideal settings are used for training. Architectures are compared in various scenarios and the best performing, with their settings, are provided. We show that from the current state-of-the-art architectures, DenseNet outperforms all others when memory is not a constraint. When memory footprint is a factor, ShuffleNet is shown to performed the best.

Keywords: convolutional neural networks; orbital angular momentum; underwater communications

1. Introduction

In 2014, Krenn et al. explored the use of machine learning (ML) to demultiplex orbital angular momentum (OAM) beam patterns for free-space optical communications [1]. Since then, ML techniques have been applied in a variety of ways to improve demultiplexing accuracy in free-space turbulent conditions [2–4].

Properties of OAM can be used in a variety of ways to enable communication. When certain OAM modes are combined together, they result in unique patterns. Our approach uses patterns to encode information at the transmitter and decode the sequences at the receiver. As decoding requires being able to differentiate between the patterns, any environmental conditions that distort patterns can introduce classification error at the receiver.

In OAM communications, turbulence and attenuation can cause significant degradation of signal integrity and lowering of the signal-to-noise ratio (SNR) [5,6]. These disturbances can displace spatial patterns, cause crosstalk, or scatter the signals such that only a portion of the original intensity distribution makes it to the receiver.

One of the unresolved questions from Ref. [2] is with regards to which, if any, of the state-of-the-art convolutional neural network (CNN) architectures performs best for OAM pattern demultiplexing in signal degrading environments. This paper explores turbulent free-space and attenuated underwater OAM optical communications with the state-of-the-art deep convolutional neural networks to answer this question.

Several data sets under varying environmental conditions are used for this effort. In free-space, three sets of data are collected at different turbulence levels. In water, four sets of data are collected at various attenuation levels. All tests are performed on specific combinations of these data sets.

Contributions of this paper include a comparison of recent, state-of-the-art CNN architecture in both turbulent free-space and attenuated underwater OAM communications. Baseline performance, inter-set performance, and parameter count are analyzed. At the end of the analysis, the best performing architectures, along with their parameters, are provided.

2. Background and Prior Art

In the following sections OAM communications, hyperparameter tuning, and an overview of some of the current state-of-the-art CNNs are covered.

2.1. Orbital Angular Momentum

Orbital angular momentum (OAM) in electric fields was discovered by Allen et al. [7]. They found that under certain conditions, the Laguerre-Gauss beam could transition from a standard plane wave propagation to a helical path. Consequently, the Gaussian-shaped distribution frequently exhibited by lasers becomes a doughnut shaped pattern when an OAM mode is adopted. The OAM azimuthal dependency is expressed by $\exp(i\ell\phi)$, where ℓ is the topological charge or mode number. When $\ell = 0$, the wavefront is a plane. When $|\ell| > 0$, the wavefront travels in a helical path, where the direction of rotation about the z-axis is controlled by the sign on ℓ . The radial distance from the z-axis to the helix is controlled by the mode number. The larger the mode number, the greater the radius.

A significant property exhibited by OAM modes is that they are orthogonal to each other [7]. Consequently, multiple OAM beams with different modes can be multiplexed together and be completely recovered at the receiver. Leveraging this property allowed Want et al. to achieve terabit data rates in ideal conditions [8]. While promising, communications in non-ideal conditions with turbulence and attenuation present hurdles in actually achieving these rates.

Various of approaches have been used to detect modes at the receiver, including conjugate mode sorting [9,10], Doppler effect measurements [11], dove prism interferometers [12], optical transformers [13], and spiral fringe counting [14]. In 2017 Doster and Watnik applied ML to the problem and found significant improvements in demultiplexing accuracy and simplification of hardware setup over existing approaches [2].

While Ref. [2] provided a proof of concept in applying CNNs to determining OAM modes, they left an exploration of the best CNN for future work. That investigation is completed here in addition to addressing other questions.

2.2. State-of-the-Art CNNs

CNNs are great for image-based applications because their convolution kernels are able to learn and differentiate shapes, colors, hues, etc. found in the training images. The unique characteristics associated with each class are learned during training and then used later for identifying classes during inference.

The deep learning revolution was accelerated with the groundbreaking results achieved by the LeNet architecture developed by LeCun [15]. The AlexNet [16] architecture later smashed previous records on the ImageNet [17] data set by combining convolution layers, fully connected layers, rectified linear units (ReLU), and dropout layers. ImageNet is a benchmark set of images including 1000 different classes and is frequently used to compare architecture performance.

Since AlexNet, a number of significant improvements have been made in CNN architectures, layers, and optimizers. Adam, for example, is an optimizer that provides adaptive learning rates [18]. As training progresses, learning rates are automatically adjusted up or down so weights do not train too slowly or diverge, because learning rates are too high.

ResNet created another revolution in the CNN architecture field by introducing the concept of special processing blocks surrounded by identity connections [19]. ResNet is able to address the problem of diminishing gradients by allowing vital image information to be made available, through the identity connections, to deeper layers of the network. ResNet is able to bypass the accuracy inflection point of the VGG architecture, where adding layers caused decreases in performance [20]. ResNet is able to continue improving performance by stacking much deeper layers, beyond which VGG performance degraded.

ResNeXt is an extension of the ResNet architecture [21]. They postulated that gains could be made through widening the architecture. They introduced the idea of cardinality, where N branches were introduced and each branch included a small number of kernels.

DenseNet provides a complimentary approach to ResNet [22]. Rather than using identity connections, DenseNet is able to feed forward feature maps from each processing block (which is composed of convolution layers and other operations). Each processing block is provided feature maps, at its input, from all previous processing blocks.

SqueezeNet is an architecture designed for small memory footprint applications [23]. It is composed of ‘Fire modules’ which include squeeze (1×1 convolutions) and expand (1×1 and 3×3 convolution) layers. It was found to perform comparable to AlexNet on the ImageNet set, but with $50 \times$ fewer parameters.

MobileNet is a series of streamlined architectures that use depth-wise separable convolutions [24]. The series was designed to be lightweight so as to be appropriate for applications such as self-driving cars, robotics, etc.

ShuffleNet is an architecture designed to be memory efficient, so as to be deployed in robotics and mobile devices [25]. While its accuracies are not competitive with architectures such as ResNet, when trained and tested against ImageNet, it has a much lighter memory footprint and can fit in small devices where the larger architectures will not. It was shown to have better accuracies on the ImageNet set than MobileNet.

SqueezeNet, MobileNet, and ShuffleNet are small, efficient architectures where AlexNet, DenseNet, ResNet, ResNeXt, and WideResNet are large but produce better results on the ImageNet benchmark data set. These architectures will be used and contrasted with each other in this work to determine which provides the best performance in the OAM communication domain. Ref. [26] provides an interesting insight that some of the state-of-the-art CNNs struggle in their ability to generalize and are in fact ‘brittle’. This warrants a careful consideration in applying the state-of-the-art to new domains. The best architecture in ImageNet classification does not necessarily mean optimal performance in another domain. These architectures are trained and analyzed to show which perform best in OAM free-space and underwater communications.

2.3. Hyperparameter Tuning

‘Hyperparameters’ are parameters that are set before training begins. Examples of these parameters include learning rates, batch size, number of training epochs, optimizer, methods for weight initialization, etc.

Learning rates control the magnitude of updates to weights during training. When learning rates are too large, training can diverge. If weights are too small, it may take a very long time to converge to a solution.

An epoch is a cycle of training where all available data has been used once to train the network. A data set is often broken into batches and trained one batch at a time. The batch size can influence how well the architecture trains. The number of epochs also influences the overall performance of the network. If the network trains for too many epochs it can over train and will not generalize well. If trained too little, then the network will not learn the unique characteristics of the information presented to it. Either case can result in poor performance.

Adam is currently among the most popular optimizers [18]. Ref. [27] found that Adam did not always perform better than other optimizers. In light of this, several additional optimizers were selected to include in the parameter search. A comparison of various optimizers was done by Ruder [28]. From their list, the following optimizers were selected for comparison: Adam, AdaMax [18], Nadam [29], and RMSProp [30].

To perform a fair comparison between the selected CNN architectures, a hyperparameter search can be used to make sure architectures have been suitably configured. Selecting a good learning rate is considered to be one of the most important hyperparameters to tune [31]. For this research, optimizers and learning rates are evaluated.

Approaches to hyperparameter tuning are an ongoing area of research. The most basic approach is a grid search, where a range of values is selected for each hyperparameter, the values are changed one at a time, and all combinations are exhaustively evaluated. This approach is computationally expensive. A more effective approach is a random search [32]. It is able to find better configurations in less time compared to grid search.

Hyperopt is a parameter tuning approach that allows searching across multiple hyperparameters in an efficient way [33]. Ref. [33] shows that Hyperopt provides an order of magnitude speed-up over Bayesian parameter tuning methods. For this research, the Hyperopt algorithm was used from the Tune package for the learning rate search [34].

3. Experiment Setup

Turbulent free-space image sets and attenuated underwater sets of data are used in this study. Free-space turbulent data is collected using the following configuration. The imagery is collected using a 635-nm 5-mW laser, a Dalsa GigE camera, several Forth Dimension Displays binary phase ferroelectric spatial light modulators (SLMs), and standard optical tools such as mirrors, pinhole filters, and diffraction order filters. The SLMs are programmed with a binary phase hologram. Simulated turbulence is also added to the holograms. MATLAB is used to generate signals that synchronize the laser, SLMs, and camera. The lab setup, originally shown in Ref. [2], is displayed in Figure 1. The images generated by the free-space camera are 512×512 pixels and subsequently cropped to 256×256 pixels. They are resized to 128×128 for computational efficiency.

Figure 1. Bench setup for free-space configuration.

The free-space OAM images consist of 32 different patterns, created through multiplexing beams passed through combinations of five different phase plates with modes $[-4, -1, 2, 5, 8]$. OAM patterns from this set are shown in Figure 2.

Figure 2. Example of OAM patterns from the free-space data set.

The free-space set is composed of three distinct data collects, where each group is collected at a specific turbulence level. Turbulence is simulated and imparted through inserting phase screens in the OAM beam path. The turbulence levels are $D/r_0 = 5$, $D/r_0 = 10$, and $D/r_0 = 15$. Where D represents the linear dimension of the SLM and r_0 is the Fried's parameter, which measures a signal's quality of transmission through the atmosphere. Thus, higher values in the ratio D/r_0 represent increased turbulence and signal displacement, which amounts to signal crosstalk. Examples of OAM patterns at different turbulence levels are showing in Figure 3. The data collects will be referred to as TB5, TB10, and TB15, where each data collect includes images from each of the 32 patterns.

Underwater data is collected using the following hardware setup. The laser source used in these measurements was a diode pumped solid state laser that operates at 532-nm and produces 5 nS pulses with 250 uJ/pulse. The intensity patterns are captured by a high performance, fast-frame-rate camera (Photron FASTCAM SA-Z). The camera is synchronized with the laser pulses at a rate of 1 kHz. As shown in Figure 4, the laser beam splits into four coherent beams and is expanded to pass through and fill vortex phase plates where an OAM phase is imparted to each beam. After leaving the phase plates, the beams are recombined using beamsplitters. The multiplexed OAM beam passes through a 1.2-m water tank and is routed to the camera using mirrors. Polyamid beads are added to the water to introduce signal scattering while small pumps agitate the water to ensure that the particles remain in suspension and homogeneously distributed. Attenuation length is measured by running a 15 mW, 532 nm CW probe laser parallel to the beam that is incident on a sensor. The OAM modes used for this configuration are [1, 4, −6, −8]. Images created by the camera for this setup are 1024×1024 pixels. The images are cropped and then resized to 128×128 pixels.

Underwater OAM images consist of 16 unique patterns derived from combinations of four different phase plates. Examples of the OAM patterns are shown in Figure 5.

The underwater set is composed of four different data collects, where each group is collected at a specific level of attenuation. Attenuation is created by adding polyamid beads designed for scattering light. Attenuation length sets are composed of levels 0, 4, 8, and 12. The sets will be referred to as AL0, AL4, AL8, and AL12. Each set includes images representing each of the 16 OAM patterns. Figure 6 shows attenuated patterns at the four levels of interest.

Figure 3. Example of different turbulence levels from the free-space data set. Column header indicates the pattern number and the row label indicates the level of turbulence. Inspecting the different levels for the OAM modes shows pattern displacement or distortion due to the turbulence.

Figure 4. Bench setup for underwater OAM communication configuration.

Figure 5. Example of OAM patterns from the underwater data set.

Each free-space and underwater data set is divided into training, validation, and test sets at a 70%/15%/15% respective split. The test sets are used only after a classifier is fully trained and its final performance metrics are gathered. The test set is often referred to as a holdout set, as it is put aside until the very end.

Training took place on a computer with a NVIDIA RTX 2080 GPU with 8 GB of RAM. The computer also has 32 GB RAM and an Intel i9 processor with 16 cores. For the CNN training and evaluation in this paper, the code is developed in Python and the ML library used is PyTorch.

Table 1 provides the total number of trainable parameters for the architectures used in this study, as reported by PyTorch.

Table 1. CNN Architecture Trainable Parameter Count.

CNN	Parameter Count
ShuffleNet	374,592
SqueezeNet	751,840
MobileNet	2,264,864
DenseNet	6,982,048
ResNeXt	2,3045,472
ResNet	23,573,600
AlexNet	57,134,944
WideResNet	66,899,808

Figure 6. Example of different attenuation levels from the underwater data set. Column header indicates the pattern number and the row label indicates the attenuation level.

4. Results

In this section, results for each of the tests are presented. In Section 4.1, results are presented for the hyperparameter search in free-space and underwater environments for each of the CNN architectures. Section 4.2 presents results for architectures trained against each data set. Finally, Section 4.3 shows results when architectures are trained with one data set and tested against other, more distorted, data sets.

4.1. Hyperparameter Tuning

One of the objectives of this paper is to identify architectures and training parameters that are best suited to the OAM pattern classification task. There are a variety of hyperparameters that effect how quickly an architecture can train as well as how well it can perform. In order to keep the parameter tuning space constrained, the following process is followed.

We first select four optimizers and do a brief case study to determine whether one does better than another. One of the more challenging data sets is selected (TB5) and the ResNeXt 50 architecture is used. ResNeXt 50 has over 23 million trainable parameters, so the architecture itself is sufficiently complex to provide some challenge while training. These selections are designed to help identify whether one optimizer works better than another. After evaluating the results, that optimizer is used for the remainder of the training.

Once the optimizer is selected, we proceed to finding learning rates for each of the architectures. To do this, we use the optimizer previously selected and then use middle complexity data sets from the underwater (AL4) and free-space (TB5) sets.

Batch size is a hyperparameter that can be changed as well. The batch size is kept the same for all architectures, minus DenseNet. This is due to the fact that memory requirements during training with DenseNet are significantly higher than the other architectures and exceed available memory in our systems.

With the optimizer, learning rates, and batch sizes all selected for each architecture and data source, we are ready to compare how well the various architectures compare against each other. It is important to highlight the fact that finding comparable hyperparameter settings is critical for objective comparison of architectures. If a learning rate, for example, is set too high or too low on one architecture it is likely to underperform relative to its peers. The underperformance in that case is not because the architecture is any worse than another, but because the hyperparameters were not properly selected. For this reason, time and effort are expended on identifying settings that will allow the architectures to perform their best.

A variety of parameters exist that influence the training of a CNN. Training a CNN generally happens over many epochs, where an epoch consists of using all training data once. In an epoch, the data set is generally divided up into smaller groups called batches. Each time a batch is passed through the CNN during training, the difference between its predictions, and the actual classes generates an error. The error is backpropagated through the CNN and is used to update the weights in the CNN. The rate at which updates are made is controlled by the learning rate. Accuracy refers to the percent of the time that the CNN assigns the correct class to an image.

Architectures are initialized with pre-trained weights from ImageNet training, and are used as the starting point for training the OAM patterns. As the ImageNet competition has 1000 classes and a fixed input size of 224×224 , the CNN input and output layers were modified for 128×128 sized input images and output dimensions for classes of 16 (underwater) and 32 (free-space).

As hyperparameter searches can quickly become computationally expensive, a two-tier approach is taken to narrow the field. The first tier is to identify an optimizer that provides the best results. The second tier it to identify the best learning rate for each architecture using the selected optimizer. For the hyperparameter study, the ResNeXt 50 architecture is used, training is limited to 5 epochs, and the TB5 data set is used. Batch size is set to 32 for DenseNet because of its memory requirements during the training process. All other architectures use batch sizes of 128.

Optimizers are selected from the set of Adam, RMSProp, AdaMax, and Nadam. The ResNeXt 50 architecture is used to train data from the TB5 free-space data set. Learning rates are selected from a range of 10^{-6} to 10^{-1} for each optimizer. A quick random search was first performed to find a few good performing starting values for each optimizer. Those values are then used as best guesses to seed a Hyperopt search to support the optimizer analysis. The Hyperopt search is allowed to run 25 iterations to identify the best performing learning rate.

Figure 7 plots accuracies achieved using the four optimizers from learning rates selected by Hyperopt search. The x -axis shows the learning rates on a log scale, while the y -axis represents the accuracy achieved on the holdout set after 5 epochs of training. It is interesting to note that all of the optimizers achieve similar peak accuracies and the overall distribution of accuracies is very similar. The primary difference being the offset of the accuracy curves relative to the learning rate. These offsets are primarily due to how the learning rates are scaled within the optimizer algorithms.

Figure 7. Optimizer accuracy to learning rate comparison using TB5 free-space data set and ResNeXt 50 architecture. Accuracies are recorded after 5 epochs of training for learning rates selected by the Hyperopt algorithm.

Figure 8 shows an accuracy curve for each optimizer over the course of 60 epochs. The TB5 data set is used for training the ReNeXt architecture in this figure. Learning rates for each optimizer are derived from the peaks from Figure 5. Figure 8 shows similar convergence rates for all of the optimizers. In this figure, Nadam reaches peak accuracy the quickest, while Adamax takes a few more epochs to reach the same level. This points to the potential advantage of using Nadam over Adamax to reduce compute time.

Figure 8. Optimizer training curve comparison using TB5 free-space data set with ResNeXt 50 architecture. Accuracies are recorded after each training epoch for a total of 60 epochs.

As Figure 8 shows similar performance between the optimizers, a simple statistical analysis is employed to make the selection of which optimizer to use. Table 2 shows the average and standard

deviations of accuracies for epochs 20–70 from Figure 8. Results in the table show that Nadam gets better average accuracy and lower standard deviation than the other optimizers. Consequently, Nadam is selected as the default optimizer for all subsequent training in this paper.

Table 2. Optimizer Averages and Standard Deviations.

	Avg	SD
Adam	97.9	0.0140
Adamax	98.7	0.0085
Nadam	98.7	0.0064
RMSProp	98.3	0.0091

With the optimizer selected, the second tier of the hyperparameter search is to identify the best learning rates for each CNN architecture. Given that there are potential differences between the free-space and underwater data sets, this search is applied to each domain to see whether there are any significant differences in learning rate selection.

Figure 9a,b show Hyperopt results for accuracy vs. learning rate. The training is limited to 7 epochs, which is sufficient to generate curves showing relative training responses for different learning rates. The region of the figures of primary interest is the rising portion of the curve as these regions suggest the most efficient place to draw learning rates from. As the learning rates increase, they also show the tipping points where training becomes unstable, weights diverge, and learning ceases. Ideal learning rates for each architecture differ from each other. This is because the number of trainable parameters and the way information flows through the architectures are different.

Figure 9. Hyperopt learning rate search using AL4 underwater and TB5 free-space image set.

For the underwater set, Figure 9a shows very similar curves for the ResNet family of architectures. ShuffleNet shows the most pronounced shift as its learning rates are shifted to the right. Differences between the architectures are more pronounced in the free-space data as shown in Figure 9b. Again, the ResNet family of architectures are similar at the same range of learning rates while ShuffleNet is also shifted far to the right in its learning rate curve. SqueezeNet appears to learn significantly slower than the other architectures. This graph turns out to be indicative of its overall performance later in the paper.

These curves provide an idea of what learning rate to use for training the architectures. Learning rates are selected moving from the left side of the curve (which begins at 10^{-7}) and are selected at approximately 95% of the peak value. This allows selection of learning rates with good efficiency, but are not so high as to create convergence problems. This learning rate selection approach was established by Ref. [35].

Final learning rates used for each architecture, in underwater and free-space environments, are derived from these figures. Results are shown in Table 3. These are the learning rates used for the rest of the training in this paper. It is interesting to note that the learning rates between the two data sets are fairly similar to each other.

Table 3. Final learning rates for architectures in underwater and free-space data sets.

Architecture	Underwater	Free-Space
AlexNet	1.2×10^{-5}	1.1×10^{-5}
DenseNet	4.2×10^{-5}	4.2×10^{-5}
MobileNet	1.2×10^{-4}	7.4×10^{-4}
ResNet	3.2×10^{-5}	3.2×10^{-5}
ResNeXt	2.4×10^{-5}	2.4×10^{-5}
ShuffleNet	4.2×10^{-4}	3.1×10^{-4}
SqueezeNet	1.5×10^{-5}	5.5×10^{-5}
WideResNet	1.4×10^{-5}	1.4×10^{-5}

Using the established learning rates, accuracy curves were generated for each architecture. This provides an initial comparison of how quickly the architectures learn and the levels that they converge to. To perform this initial study, only one AL and one TB data set was used to provide a high-level view to compare the architectures. The middle set was selected for each environment to provide enough attenuation and turbulence to highlight differences between architecture performance.

Figure 10a shows accuracy per training epoch curves for the underwater AL4 data set for each architecture (at the learning rates indicated in Table 3). It is apparent from the curves that, over time, all of the architectures achieve fairly similar accuracies. While there are some differences in the initial slope of the accuracy curves, they all wind up converging at a high accuracy.

Figure 10b shows accuracy per training epoch curves for the free-space TB5 data set for each architecture. Most of the architectures settle in at approximately the same end accuracy, the lone difference being SqueezeNet.

Figure 10. Accuracy training curves for AL4 underwater and TB5 free-space OAM sets.

Aside from SqueezeNet, there does not appear, at this point, to be a great deal of difference from one architecture to another when using the AL4 (underwater) and TB5 (free-space) data sets. With hyperparameters selected, the architectures are ready for training against the data sets.

4.2. Baseline, Intra-Set Tests

Baseline performance of the underwater data sets (AL0, AL4, AL8, AL12) and free-space sets (TB5, TB10, TB15) is established in this section. In establishing baseline performance, the focus is placed on training an architecture with one data set and testing it the corresponding holdout set. Later, an architecture will be trained with one data set (TB5 for example) and then the holdout sets from TB10 and TB15 (inter-set testing) will be used to explore how well the architecture is able to generalize on data outside the training set. Percent accuracy is used at the metric for comparing relative performance of the different architectures.

Table 4 shows the baseline results for architectures trained with the underwater attenuated sets. As a whole, the architectures perform very well. Most accuracies achieved are 100%, or close to it. The only outlier is SqueezeNet on the AL12 set.

Table 4. Architecture baseline performance with underwater sets.

	AL0	AL4	AL8	AL12
SqueezeNet	99.5	99.4	99.4	98.8
AlexNet	99.4	100.0	99.4	99.4
ShuffleNet	100.0	99.4	100.0	100.0
DenseNet	100.0	100.0	100.0	99.4
ResNeXt	100.0	100.0	100.0	99.4
MobileNet	100.0	100.0	100.0	100.0
ResNet	100.0	100.0	100.0	100.0
Wide ResNet	100.0	100.0	100.0	100.0

Table 5 shows the baseline results for architectures trained with the free-space turbulent sets. This table includes a column that averages the results to help with sorting the architectures. The results in these tables are sorted by accuracy and show that the complexity imposed by turbulence, effects accuracy more than attenuation does. This table shows that AlexNet and SqueezeNet struggled the most. DenseNet appears to provide the best performance with the free-space data sets.

Table 5. Architecture baseline performance with free-space sets.

	TB5	TB10	TB15	Avg
SqueezeNet	96.9	90.4	88.3	91.9
AlexNet	99.7	99.6	83.0	94.1
ShuffleNet	99.9	99.9	100.0	99.9
MobileNet	99.8	100.0	100.0	99.9
Wide ResNet	99.8	100.0	100.0	99.9
ResNet	99.9	100.0	100.0	99.9
ResNeXt	99.9	100.0	100.0	99.9
DenseNet	100.0	100.0	100.0	100.0

Table 6 shows an example of the amount of time it took to train each architecture. The table shows the number of training epochs as well as the amount of time it took to train for the specified number of epochs. The training loop had a maximum number of training epochs, but was allowed to terminate early when a specific level of accuracy (99.9%) had been achieved on the validation set. Most of the architectures trained quickly in terms of epochs and overall time. AlexNet and SqueezeNet took the longest amount of time to train while yielding the poorest results. DenseNet and the ResNet family required the fewest training epochs and took a comparable amount of time to train.

Table 6. Training epochs and time for the TB5 free-space set.

Architecture	Epochs	Time (sec)
AlexNet	80	636
DenseNet	5	96
MobileNet	20	130
ResNet	6	110
ResNeXt	7	78
ShuffleNet	21	80
SqueezeNet	80	344
Wide ResNet	6	90

4.3. Inter-Set Performance Analysis

Section 4.2 shows that most of the architectures perform well when tested with the holdout test sets from the original data set. In real environments the trained classifiers are likely to be presented with images that have been distorted by larger turbulence and attenuation than what was present in the training set. The tests in this section explore the architectures and how well they perform when presented with images outside of their training set. For example, how well does an architecture trained with the AL0 data set classify attenuated images from the AL4, AL8, and AL12 holdout sets?

For this analysis, both underwater and free-space data sets are evaluated. For the underwater sets, the AL0 and the AL0-4 trained architectures are used. These architectures are evaluated against the AL0, AL4, AL8, and AL12 holdout sets. For the free-space data sets, the TB5 and TB5-10 trained architectures are used. These architectures are evaluated with the TB5, TB10, and TB15 holdout sets. In both cases, the results of interest are with data sets that fall outside the training sets. In the following tables, the results are ordered by ascending accuracies.

Table 7 includes results from architectures that have been trained with the AL0 data set. The four columns include accuracies from AL0, AL4, AL8, and AL12 holdout sets. In looking at the performance of the AL4 test set, DenseNet and Wide ResNet give the best accuracies at 63.7% and 81.2% respectively.

Table 7. Underwater AL0 inter-set test. Architectures trained on AL0 and tested against all AL data sets.

	AL0	AL4	AL8	AL12
MobileNet	100.0	29.4	3.1	3.7
AlexNet	99.4	44.4	18.8	10.6
SqueezeNet	97.5	45.0	23.7	7.5
ShuffleNet	100.0	45.6	16.3	6.9
ResNet	100.0	46.9	13.8	15.6
ResNeXt	100.0	51.2	13.1	12.5
DenseNet	100.0	63.7	18.1	5.0
Wide ResNet	100.0	81.2	7.5	5.6

In evaluating architectures trained with the combined AL0 and AL4 data sets, Table 8 shows results ordered according to ascending results for the AL8 data set. DenseNet and ShuffleNet take the lead spots with 80.0% and 84.4% respectively.

Table 8. Underwater AL0-4 inter-set test. Architectures trained on AL0-4 and tested against all AL data sets.

	AL0	AL4	AL8	AL12
ResNet	100.0	99.4	46.3	19.4
AlexNet	99.4	100.0	58.8	13.1
MobileNet	100.0	100.0	61.3	13.1
Wide ResNet	99.4	100.0	70.0	23.1
SqueezeNet	100.0	99.4	71.3	17.5
ResNeXt	100.0	99.4	79.4	36.9
DenseNet	100.0	100.0	80.0	40.0
ShuffleNet	100.0	100.0	84.4	45.0

Tables 9 and 10 have similarly organized results for free-space data sets. ResNet and DenseNet (97.8% and 97.9%) have the best results for TB10 in Table 9, while DenseNet and ResNet (81.8% and 84.8%) take the lead spots in Table 10.

Table 9. Free-space TB5 inter-set test. Architectures trained on TB5 and tested against all TB data sets.

	TB5	TB10	TB15
SqueezeNet	98.6	83.9	62.4
AlexNet	97.4	86.4	58.0
ShuffleNet	97.6	92.2	75.9
MobileNet	98.8	93.9	77.0
Wide ResNet	98.6	94.5	82.2
ResNeXt	98.9	95.4	81.4
ResNet	98.8	97.8	87.8
DenseNet	99.6	97.9	85.5

Table 10. Free-space TB5-10 inter-set test. Architectures trained on TB5-10 and tested against all TB data sets.

	TB5	TB10	TB15
AlexNet	97.4	82.0	59.8
SqueezeNet	98.6	83.2	61.1
MobileNet	97.6	91.3	74.0
Wide ResNet	98.6	95.1	75.2
ShuffleNet	99.0	93.5	78.5
ResNeXt	97.9	94.1	79.3
DenseNet	99.0	96.7	81.8
ResNet	98.9	96.9	84.8

5. Discussion

For this effort, three lightweight architectures (MobileNet, ShuffleNet, and SqueezeNet) and five heavyweight architectures (AlexNet, DenseNet, ResNet, ResNeXt, and Wide ResNet) are evaluated.

From Section 4.2, the free-space data set provides interesting insight in performance differences. Table 5 shows DenseNet performing the best; however, the ResNet family of architectures follow very closely in achieved accuracies.

The main conclusion from the underwater results in Table 4 is to avoid the SqueezeNet architecture. All other architectures seem to give comparable performance.

Another observation in comparing Tables 4 and 5 is the difference in accuracies between the two. The underwater data sets have great performance, even with the high signal-to-noise ratio. Why do the CNNs perform so well with the attenuated versus the turbulent images? The answer is likely due to the fact that, the overall shape of the attenuated images remains constant, where turbulence causes displacement of the image patterns. The CNNs have to work harder to learn many potential patterns that belong to a specific OAM set. In addition, the turbulence patterns have twice the number of image patterns to learn.

Transitioning to out-of-training-set testing provides even further insight into architecture robustness. Tables 7–10 have one consistent top performer, DenseNet. DenseNet took about the same amount of time and number of epochs to train as the ResNet family. It was also smaller than the ResNet family of architectures by at least a factor of 3. Thereofre, it was a consistent top performer, while also having significantly fewer parameters than many of the close performing architectures.

For a resource constrained system that cannot support the size of DenseNet or other heavy-weights, what is the next best performer? The options come down to ShuffleNet, SqueezeNet, and MobileNet. Interestingly, in reviewing tables from Sections 4.2 and 4.3, ShuffleNet consistently performed as well as or better than the other architectures most of the time. Additionally, it is considerably smaller than the other two architectures.

6. Summary and Conclusions

This paper set out to evaluate OAM transmitted turbulent, free-space and attenuated, underwater images. The specific purpose is to identify, which, if any, of the state-of-the-art CNN architectures performs best in the two environments.

Steps were taken, using a parameter search, to identify the best performing optimizer and near-ideal learning rates for training the architectures. Four underwater and three free-space data sets were then used to train each architecture.

With the trained architectures, test sets were presented from each data set to evaluate their classification accuracy. For the baseline testing (no out of set images), architectures with the underwater set appeared to perform at a level comparable to each other. The lone outlier was SqueezeNet. The turbulent free-space images, however, presented a challenge sufficient to start separating out varying levels of performance. DenseNet was the best performer from this set of tests.

Advancing to the inter-set testing, additional performance differences emerged. These tests took architectures trained with a data set, such as AL0, and then that trained architecture was presented holdout data from more attenuated data sets. This was repeated with the turbulent, free-space data as well.

In the end, it was found that DenseNet consistently performed the best, or as a close second, all the time. This finding is interesting considering its parameter count is at least a factor of three less than the other architectures that performed well. This result implies that DenseNet generalizes well during training.

For systems that are more resource constrained, three architectures with lower parameter count were tested: ShuffleNet, SqueezeNet, and MobileNet. In evaluating the results, ShuffleNet consistently performed better than or was close to the performance of the other two. It is also has considerably fewer parameters than the others. For a resource constrained system, ShuffleNet would be the clear choice due to its low parameter count and competitive accuracy.

For the turbulent, free-space data sets DenseNet was trained for 5 epochs with the learning rate set to 4.2×10^{-5} , while ShuffleNet was trained for 21 epochs with the learning rate set to 3.1×10^{-4} . Their respective batch sizes were 32 and 128. Nadam was used as the optimizer used in training all architectures.

For the attenuated underwater data sets, DenseNet was trained for 5 epochs with a learning rate of 4.2×10^{-5} and a batch size of 32. ShuffleNet was trained for 21 epochs with the learning rate set to 4.2×10^{-4} with batch size of 128.

This work provides an in-depth comparison of state-of-the-art, deep convolutional neural networks in turbulent free-space and attenuated underwater environments. It shows which architectures provide the most robust performance for degraded conditions outside of the training set. For low memory systems, ShuffleNet performed the best. For higher capacity systems, DenseNet consistently performed the best. Training parameters for both architectures are provided. Future work includes evaluating architectures in environments that include both attenuation and turbulence. Also, it may be that a better architecture, with fewer parameters still exists that is best suited for OAM images. A network architecture search could be used to identify a better architecture for classifying OAM images.

Author Contributions: Contributions by the authors include the following breakout: conceptualization, P.L.N. and A.T.W.; Methodology, N.S.F.; Software, P.L.N.; Validation, P.L.N. and N.S.F.; Formal analysis, P.L.N.; Investigation, P.L.N.; Resources, A.T.W.; Data curation, J.R.L.; Writing—original draft preparation, P.L.N.; Writing—review and editing, A.T.W.; Supervision, K.P.J.; Project administration, K.P.J.; Funding acquisition, K.P.J. All authors have read and agreed to the published version of the manuscript.

Funding: We would like to acknowledge support for this project from the US Naval Research Laboratory through a 6.1 base program for orbital angular momentum and a 6.2 base program for underwater communications.

Acknowledgments: Thank you to the research group of Eric Johnson for fabrication of the vortex phase plates.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Networks
ML	Machine Learning
OAM	Orbital Angular Momentum
SLM	Spatial Light Modulator
SNR	Signal to noise ratio

References

1. Krenn, M.; Fickler, R.; Fink, M.; Handsteiner, J.; Malik, M.; Scheidl, T.; Ursin, R.; Zeilinger, A. Communication with spatially modulated light through turbulent air across Vienna. *New J. Phys.* **2014**, *16*, 113028. [\[CrossRef\]](#)
2. Doster, T.; Watnik, A.T. Machine learning approach to OAM beam demultiplexing via convolutional neural networks. *Appl. Opt.* **2017**, *56*, 3386–3396. [\[CrossRef\]](#)
3. Lohani, S.; Glasser, R.T. Robust Free Space OAM Communications with Unsupervised Machine Learning. In *Frontiers in Optics*; Optical Society of America: Washington, DC, USA, 2019. [\[CrossRef\]](#)
4. Rostami, S.; Saad, W.; Hong, C.S. Deep Learning With Persistent Homology for Orbital Angular Momentum (OAM) Decoding. *IEEE Commun. Lett.* **2020**, *24*, 117–121. [\[CrossRef\]](#)
5. Nichols, J.M.; Emerson, T.H.; Cattell, L.; Park, S.; Kanaev, A.; Bucholtz, F.; Watnik, A.; Doster, T.; Rohde, G.K. Transport-based model for turbulence-corrupted imagery. *Appl. Opt.* **2018**, *57*, 4524–4536. [\[CrossRef\]](#) [\[PubMed\]](#)
6. Cochenour, B.; Dunn, K.; Laux, A.; Mullen, L. Experimental measurements of the magnitude and phase response of high-frequency modulated light underwater. *Appl. Opt.* **2017**, *56*, 4019–4024. [\[CrossRef\]](#)
7. Allen, L.; Beijersbergen, M.W.; Spreeuw, R.J.C.; Woerdman, J.P. Orbital angular momentum of light and the transformation of Laguerre-Gaussian laser modes. *Phys. Rev. A* **1992**, *45*, 8185–8189. [\[CrossRef\]](#)
8. Wang, J.; Yang, J.Y.; Fazal, I.; Ahmed, N.; Yan, Y.; Huang, H.; Ren, Y.; Yue, Y.; Dolinar, S.; Tur, M.; et al. Terabit free-space data transmission employing orbital angular momentum multiplexing. *Nat. Photonics* **2012**, *6*, 488–496. [\[CrossRef\]](#)
9. Gibson, G.; Courtial, J.; Padgett, M.J.; Vasnetsov, M.; Pas’ko, V.; Barnett, S.M.; Franke-Arnold, S. Free-space information transfer using light beams carrying orbital angular momentum. *Opt. Express* **2004**, *12*, 5448–5456. [\[CrossRef\]](#) [\[PubMed\]](#)
10. Mair, A.; Vaziri, A.; Weihs, G.; Zeilinger, A. Entanglement of the orbital angular momentum states of photons. *Nature* **2001**, *412*, 313–316. [\[CrossRef\]](#) [\[PubMed\]](#)
11. Lavery, M.P.J.; Speirits, F.C.; Barnett, S.M.; Padgett, M.J. Detection of a Spinning Object Using Light’s Orbital Angular Momentum. *Science* **2013**, *341*, 537–540. [\[CrossRef\]](#)
12. Leach, J.; Padgett, M.J.; Barnett, S.M.; Franke-Arnold, S.; Courtial, J. Measuring the orbital angular momentum of a single photon. *Phys. Rev. Lett.* **2002**, *88*, 257901. [\[CrossRef\]](#) [\[PubMed\]](#)
13. Lavery, M.P.J.; Berkhout, G.C.G.; Courtial, J.; Padgett, M.J. Measurement of the light orbital angular momentum spectrum using an optical geometric transformation. *J. Opt.* **2011**, *13*, 064006. [\[CrossRef\]](#)
14. Soskin, M.S.; Gorshkov, V.N.; Vasnetsov, M.V.; Malos, J.T.; Heckenberg, N.R. Topological charge and angular momentum of light beams carrying optical vortices. *Phys. Rev. A* **1997**, *56*, 4064–4075. [\[CrossRef\]](#)
15. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [\[CrossRef\]](#)
16. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [\[CrossRef\]](#)
17. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009.
18. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
19. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.

20. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
21. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated Residual Transformations for Deep Neural Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5987–5995.
22. Huang, G.; Liu, Z.; Maaten, L.V.d.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269. [[CrossRef](#)]
23. Iandola, F.N.; Moskewicz, M.W.; Ashraf, K.; Han, S.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1 MB model size. *arXiv* **2016**, arXiv:1602.07360.
24. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arxiv:1704.04861.
25. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.
26. Azulay, A.; Weiss, Y. Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv* **2018**, arXiv:1805.12177.
27. Wilson, A.C.; Roelofs, R.; Stern, M.; Srebro, N.; Recht, B. The Marginal Value of Adaptive Gradient Methods in Machine Learning. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 4151–4161.
28. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.
29. Dozat, T. Incorporating Nesterov Momentum into Adam. In Proceedings of the ICRL 2016, San Juan, Puerto Rico, 2 May 2016.
30. Tieleman, T.; Hinton, G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. *COURSERA Neural Netw. Mach. Learn.* **2012**, 4, 26–31.
31. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; The MIT Press: Cambridge, MA, USA, 2017.
32. Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **2012**, 13, 281–305.
33. Bergstra, J.; Yamins, D.; Cox, D.D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In Proceedings of the 30th International Conference on International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; Volume 28, pp. I-115–I-123.
34. Liaw, R.; Liang, E.; Nishihara, R.; Moritz, P.; Gonzalez, J.E.; Stoica, I. Tune: A Research Platform for Distributed Model Selection and Training. *arXiv* **2018**, arXiv:1807.05118.
35. Smith, L.N. Cyclical Learning Rates for Training Neural Networks. In Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 24–31 March 2017.

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).