Vegetation Coverage in Marsh Grass Photography Using Convolutional Neural Networks

Lucas Welch
University of North Florida
1 UNF Drive, Jacksonville, FL 32224
n00797216@unf.edu

Xudong Liu
xudong.liu@unf.edu

Ikdika Kahanda
indika.kahanda@unf.edu

Sandeep Reddivari
sandeep.reddivari@unf.edu

Karthikeyan Umapathy
k.umapathy@unf.edu

Abstract

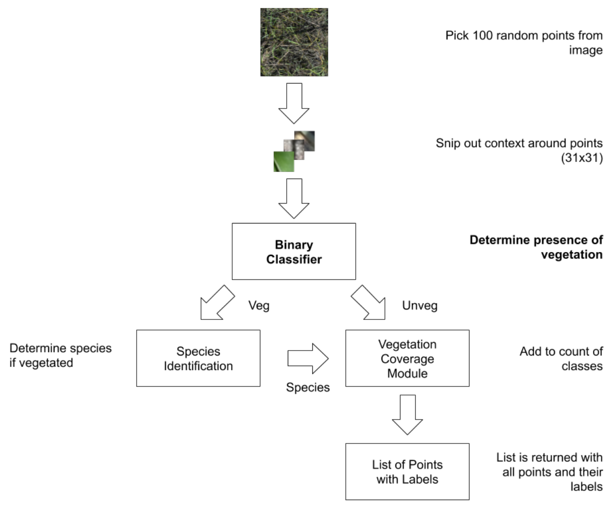
Vegetation monitoring is one of the major cornerstones of environmental protection today, giving scientists a look into changing ecosystems. One important task in vegetation monitoring is to estimate the coverage of vegetation in an area of marsh. This task often calls for extensive human labor carefully examining pixels in photos of marsh sites, a very time-consuming process. In this paper, aiming to automate this process, we propose a novel framework for such automation using deep neural networks. Then, we focus on the utmost component to build convolutional neural networks (CNNs) to identify the presence or absence of vegetation. To this end, we collect a new dataset with the help of Guana Tolomato Matanzas National Estuarine Research Reserve (GTMNERR) to be used to train and test the effectiveness of our selected CNN models, including LeNet-5 and two variants of AlexNet. Our experiments show that the AlexNet variants achieves higher accuracy scores on the test set than LeNet-5, with 92.41% for a AlexNet variant on distinguishing between vegetation and the lack thereof. These promising results suggest us to confidently move forward with not only expanding our dataset, but also developing models to determine multiple species in addition to the presence of live vegetation.

Introduction

Around the world, humans are having a large impact on the environment. These impacts are varied and are being felt by our society more every day. Because of these pressures, it is important to monitor the vegetation community of a habitat. These vegetation communities are easily disturbed by both sea level rise and habitat loss (Warren and Niering 1993). At Guana Tolomato Matanzas National Estuarine Research Reserve (GTMNERR)[1], researchers monitor the percent cover and species composition of several marsh sites surrounding the city of St. Augustine, Florida. To do this, they first obtain one meter-square images of the marshland with a high resolution camera (Bacopoulos, Tritinger, and Dix 2019). Determining the percent vegetation cover of the image requires a volunteer or researcher to label a set of randomly chosen points and tally the total number of each category (unvegetated and one of five vegetated categories) in order to obtain the percent cover of each image. Because this must be done manually and is labor-intense, the reserve is limited in how much area it is able to monitor. As such, it is our goal to automate this process by first creating a model to determine whether a small randomly chosen snippet is vegetated or not. This will eliminate much of the labor requirement for the vegetation monitoring program at GTMNERR. This will allow for more images to be processed by GTMNERR, increasing their productivity. In this paper we describe our approach to solve a relaxed version of this problem—determining the presence or absence of vegetation at one of these points. This model can be simpler and serve as a proof of concept for the idea of determining vegetation density. The model best suited for this task will, at a later time, be coupled with a model to determine the species contained on a vegetated point; together, these two models will determine the class to which a point belongs.

This task, however, is not without precedent, as machine learning techniques have become popular in ecology—image recognition having wide ranging applications in such a visual field. For instance, Mihail et al. 2018 develop a model that is able to segment an image for Spanish moss and use these image segmentations to attempt to measure the density of Spanish moss in the images. In the realm of marine ecology, it has also become commonplace for image recognition models to be used for coral identification on reefs. For instance, Marcos et al used supervised learning to train a feedforward neural network to distinguish between live coral, dead coral, and bare seabed (sand) (Marcos, Soriano, and Saloma 2005). Additionally, work has been done on Serengeti camera traps (hidden cameras designed to candidly photograph wildlife) with the goal of identifying and counting fauna that are photographed with the traps (Tabak et al. 2019). This model had the added feature of distinguishing between a number of behaviors such as eating, and sleeping. These concepts, however, have not yet been applied to marsh monitoring. As such, this paper presents the first model in a pipeline of mod-

[1]`https://gtmnerr.org/`



Pick 100 random points from image
Snip out context around points (31x31)
Binary Classifier
Determine presence of vegetation
Veg
Unveg
Determine species if vegetated
Species Identification
Species
Vegetation Coverage Module
Add to count of classes
List of Points with Labels
List is returned with all points and their labels

Figure 1: A flowchart representing the major components of the vegetation density pipeline. The pipeline begins with a full-size image before picking 100 random points and snipping the context around them (31x31). They are then passed into the described model, which will classify and pass them along to either the next classifier or tallier. The total number of each species for the 100 points is then printed to a CSV file.

ules to carry out the task of automating the vegetation monitoring program at GTMNERR.

To this end, our goal is to automate the process of determining the vegetation density in given marsh grass photography. In Figure 1, we describe a pipeline to achieve our goal. The first component is a random number generator that, given the height and width of an image, will choose 100 random coordinates. The next module will then cut out a snippet of 31 pixels by 31 pixels around each point, which will then be passed into the binary classifier model, which will determine if the central pixel contains vegetation. If the pixel is unvegetated, the output is used by a tallying module to increment the number of bare points by one. If the pixel is determined to have vegetation, however, it is passed to another module that will determine the species of the vegetation before being tallied as a member of that species by the tallying module. After all 100 random points have been classified, the tallying module will record the number of each species found in the image, along with the images name. This is taken as the percent cover of each species.

In this paper, we will be discussing the data set that was designed for this novel application. We will also discuss the binary classifier for distinguishing between vegetation and the lack thereof. Afterward, we discuss the results of the training and compare the various models before addressing the work that is still be done on the additional modules as well as the wrapper program that will be implemented to track the number of points of each class contained in a given image.

Preliminaries

In this work, we present a multi-class vegetation dataset of Florida marshland images and demonstrate the effectiveness of LeNet-5 and AlexNet on predicting whether an image contains vegetation or not for our dataset.

The images we collected are snippets of the one meter-squares, and these snippets are labeled with six classes: *bare*, *Spartina alterfloris*, *Batis maritima*, *Juncus roemerianus*, *Avicennia germinans*, and *Sarcocornia perennis*. The point of interest that is being labeled is at the center of these snippets. They are 31x31 images that are full-color three channel, and they are normalized by dividing the RGB values by 255. The central points are chosen from any point more than 15 pixels from the edge of the image. The size of 31x31 was chosen for a window size because of the desire to eliminate the need to work with padding. However, the desire to eliminate padding means that central points cannot be less than one half the length of the snippet from the edge of the image. Because of this, the larger the snippets are, the less area one has to select from for random points. As such, we must find a balance between how large of a subimage we pass into the model and how many possible random points we have from which to choose. As such, 31x31 was the first value we decided to test. Points were chosen for the dataset such that windows could be as large as 101x101. This does not necessarily mean that the final pipeline will only choose points more than 50 pixels from the border, however. For the time being, 31x31 will be used with the plan to test several snippet sizes in the future (see Future Work section). The images are passed into the two models, LeNet-5 and AlexNet, as arrays of dimensions 31x31x3.

Both models in this paper are Convolutional Neural Networks (CNNs) that have previous appearances in literature (LeNet-5 and AlexNet). We also manipulate AlexNet further to reduce the amount of time spent training the network. In the following sections, we discuss how we implement the models to adapt to our learning tasks and our data. The models discussed here were trained on loss and evaluated using accuracy. Because it is a binary classification problem being solved, the accuracy is defined as $(number\ of\ true\ positives + number\ of\ true\ negatives)\ /\ total\ data\ set\ size$.

LeNet-5

LeNet-5 is a simple CNN that was designed and implemented for optical character recognition on the MNIST dataset of images of 32x32x1(LeCun et al. 1998). Because our images are of a different size, we adopt a slight variation of the original LeNet-5 model. The details of the LeNet-5 model we adapt are included in Table 1. There are two differences between our model and the original model. One is the filter size used in the first convolutional layer (C1) is 4x4, instead of 5x5 in standard LeNet-5. Because of this tweak in the first convolutional layer, it produces the output of size 28x28, same with the standard model, to ensure the rest of the hidden layers are the same as well. The other difference is the size of the output layer, for which we have 2 values for the binary classification task, as opposed to 10 values for the single digits. From Table 1, we calculate the number of parameters to be learned is 59,964.

AlexNet and mAlexNet

Similar to LeNet-5, AlexNet also is a convolutional neural network with numerous convolution and pooling layers(Krizhevsky, Sutskever, and Hinton 2012). It is a more complex model than LeNet-5, primarily attributed to the fact that AlexNet has more convolutional layers with more feature maps. However, the size of the input image originally used for AlexNet was 224x244 with three color channels. Because our input image is only 31x31, we had to make adjustments to the stride and receptive field sizes of the layers in the network in order to fit the model to the size of the image. As such, while the number of feature maps and number of units in fully connected layers were kept the same, we adjusted kernel size and stride to work with the image size we have. The hidden layers, much like LeNet-5, consist of an alternating pattern of convolutions and pooling layers. However, for AlexNet, max pooling is used instead of average pooling. As with our LeNet-5 variant, we resort to a different implementation of AlexNet because of the different input size. Table 2 provides the details of our adopted AlexNet model, where we see the differences are not only in the first convolutional layer and output layer, but also in the intermediate layers so as to fit our input size. From the parameters provided in Table 2, we calculate the number of parameters to be 24,271,968. Clearly, there are more parameters for AlexNet than for LeNet-5; this is caused mostly by the much deeper feature space of AlexNet compared to its counterpart, potentially enabling AlexNet to solve more difficult learning problems.

Due to this sheer large number of trainable parameters, we observe extensive training time shown later in our experiment sections. Consequently, we also propose to experiment another model, a modified variant of AlexNet, for which we call *mAlexNet*, with the two fully connected layers of 100 neurons instead, resulting in about 3.4 million parameters, a very considerable decrease from AlexNet. We include such differences between AlexNet and mAlexNet in Table 2.

The Dataset

The images used for the creation of the dataset were provided by GTMNERR in St. Augustine, Florida. The image set from which our data was obtained consists of 811 one square-meter photographs of mash grass known as "photoquadrats." These photoquadrats, each of which is of dimension ranging from 1653x1666 to 3268x2830, are used to obtain data in a per-square-meter fashion that allows for easy interpretation (such as density of a species per square meter). Because the program currently used by GTMNERR does not record the coordinates for the random pixels chosen, we were not able to use the work previously done by GTMNERR in the creation of our dataset. As such, we had to create our own set of points for labeling. To do this, we chose 100 random pixels from each image that fit the criteria of being at least 50 pixels away from the edge of the image, so, when passing an image into our algorithms, we would not have to pad the image in order for it to fit our input vector for the neural network—even when we begin to try different snippet sizes.

As far as can be seen, there is no publicly available data set for this problem. Aside from our results in this paper, this provides an opportunity for us to contribute a marsh dataset to the public to allow others to attempt this problem. The data set consists of a set of 811 different photoquadrat images and a list of 57,372 points with the image the point is found in, the coordinates for the point, and a label representing the class to which the point belongs.

After generating a set of pixels to use for the dataset, we then had to label each one for use as our ground truth. This was done with a group of volunteers who are trained university students. The dataset consists of points that are classified into six different categories: *bare* or *unvegetated*; *Spartina*, which is the dominant species in the marsh; *Juncus*; *Batis*; *Avicennia*; and *Sarcocornia*. The latter four are less common than *Spartina*, and, as such, are less represented in the data.

When analyzing our data, we see the unvegetated class is the most prevalent with over 71.9% of the images, compared to 28.1% being vegetated. This is, however, to be expected, for every site that was sampled, while it never contained all species, contained bare areas. We also see the lack of data for *Avicennia* and *Sarcocornia*, which combine for only 0.4% of the dataset. This is drastic compared to *Spartina*, *Batis*, and *Juncus*, which have frequencies of 18.3%, 6.9%, and 2.4% respectively. This disproportion may be challenging to training an effective species identifier; nonetheless, it is not much of a concern, for we focus on learning a binary classifier to separate unvegetated from vegetated images in this work.

Experimentation and Analysis

To this end, we set off to experiment with the three aforementioned CNN models—LeNet-5, AlexNet, and mAlexNet—for our marsh dataset to show their effectiveness of deciding whether the area in a marsh image is vegetated or not.

To solve this binary classification problem for each model, we combine all five species classes into one "vegetated" class. We then carry out a stratified test-train split with a ratio of 80% training and 20% test set. We then place the test set aside until the end when a best model is decided. We split the training set, through a simple random sample, into 100 near-equally sized buckets to be used for constructing a learning curve.

Learning curves for the three CNNs are constructed in the same manner. Using a hold-out method, we assign the first bucket to the training set and the other

Table 1: Structure and number of parameters of each layer in LeNet-5 model for 31x31 image. Parameters have been tweaked to accommodate the different size of the input.

Layer Type	Size	Feature Maps	Kernel Size	Stride	Activation	Parameters
Input	31x31	3	—	—	—	—
Convolution	28x28	6	4x4	1	ReLU	294
Avg Pooling	14x14	6	2x2	2	ReLU	—
Convolution	10x10	16	5x5	2	ReLU	1216
Avg Pooling	5x5	16	2x2	2	ReLU	—
Convolution	1x1	120	5x5	1	ReLU	48120
Fully Connected	84	—	—	—	ReLU	10164
Output	2	—	—	—	Softmax	170
Total						59964

Table 2: Structure and number of parameters of each layer in AlexNet model for 31x31 image. Parameters have been tweaked to accommodate the different size of the input. Included are the number of trainable parameters for our downsized AlexNet model as well, separated by a "/"

Layer Type	Size	Feature Maps	Kernel Size	Stride	Activation	Parameters
Input	31x31	3	—	—	—	—
Convolution	28x28	96	4x4	1	ReLU	4704
Max Pooling	13x13	96	3x3	2	ReLU	—
Convolution	11x11	256	3x3	1	ReLU	221,440
Max Pooling	9x9	256	3x3	1	ReLU	—
Convolution	7x7	384	3x3	1	ReLU	885,120
Convolution	5x5	384	3x3	1	ReLU	1,327,488
Convolution	3x3	254	3x3	1	ReLU	878,078
Max Pooling	2x2	254	2x2	1	ReLU	—
Fully Connected	4096/100	—	—	—	ReLU	4,165,632/101,700
Fully Connected	4096/100	—	—	—	ReLU	16,781,312/10,100
Output	2	—	—	—	Softmax	8194/202
Total						24,271,968/3,428,832

99 to the validation set. We then train a model using the training set and validate it on the validation set. The number of buckets used for the training set for each model is incremented by one until 20 buckets are being used for training—at which point the number of buckets being used for training will be incremented by 10 until we are using 90 buckets for training, at which point the remaining 10 buckets are used for validation.

Thereafter, the 100 buckets are merged into 10, buckets 1–10 into fold 1, etc., for a final 10-fold cross validation, when a best model is selected to test on the test set to report its predicting accuracy.

In the following, we present our experimental results and analysis in the order of learning curves and then cross validations before testing results unveiled.

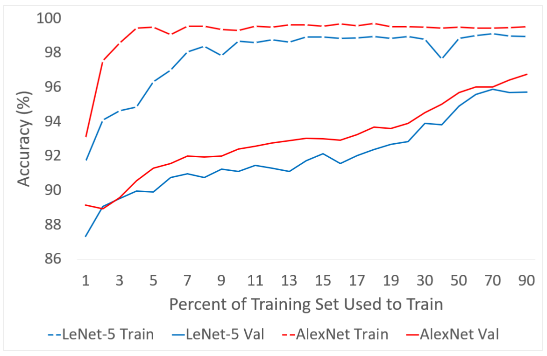
Learning Curves

LeNet-5 vs. AlexNet: We start with training LeNet-5 and AlexNet models to see their learning curves and testing accuracies in cross validations.

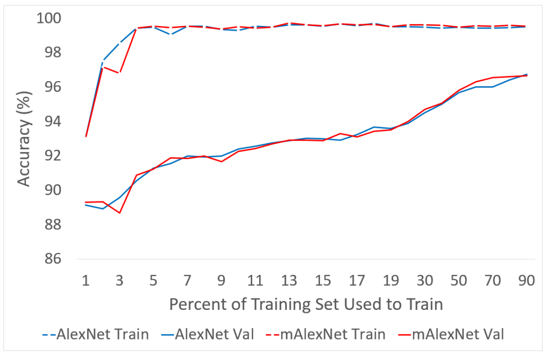
The instances in the learning curves are done five iterations and the averages are reported in 2a, where we present accuracies of models relevant to experience that is the set of training examples. We see that, though both LeNet-5 and AlexNet exhibit underfitting, AlexNet has a clear advantage over LeNet-5. For AlexNet, we have a peak validation accuracy of over 96.74% versus LeNet-5's of 95.73%, over 1% more accurate.

However, the training time for AlexNet turns out overwhelming for large numbers of training examples. We plot the computational time over training sample sizes, blue, in 2d. This training time varies from 340 minutes to train an AlexNet model when a training set 1% of the total training set size, to 1022 minutes to train a model for 90%. These large training times result in taking, on average, 9.9 days for each iteration of the learning curve, our full 5 iteration learning curve for the original size AlexNet taking very near 50 days to generate. This time is too excessive and a faster version of AlexNet is desirable. Consequently, we introduce a modified variant, mAlexNet, with same convolutional and pool layers but with much fewer trainable parameters. Next, we show mAlexNet is highly comparable to AlexNet in learning curves, yet train in noticeably shorter time.

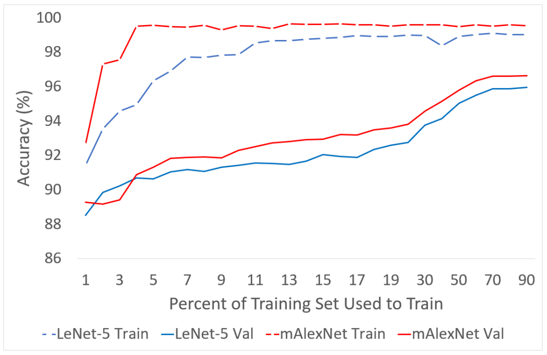
AlexNet vs. mAlexNet: Learning curves for AlexNet and its modified counterpart are shown in 2b. Clearly, the two models generally are close to identical on



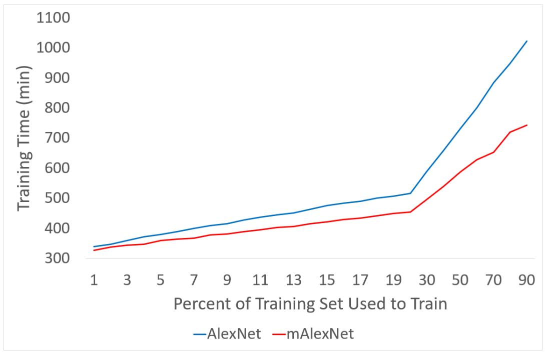
(a) LeNet-5 vs. AlexNet over 5 iterations



(b) AlexNet vs. mAlexNet over 5 iterations



(c) LeNet-5 vs. mAlexNet over 10 iterations



(d) Comparing training time

Figure 2: Learning curves comparing our three models—LeNet-5 and both AlexNet models—over 5 or 10 iterations. Figure 2d is also included , which compares the time to train a model of both AlexNets using a given percentage of the training set.

Table 3: Table denoting the execution time in our three models. This was done by tracking the time to pass all examples in the dataset into the model. We then calculate the average time per snippet and the average time per 100 snippets (number of snippets in an image assessment). All times are in seconds.

Model	Dataset	Snippet	Image
LeNet-5	3.786	6.60E-5	0.006
AlexNet	40.718	7.10E-4	0.071
mAlexNet	30.24	5.27E-4	0.053

both training and validation accuracies. The two models are very similar in how they behave too. Moreover, mAlexNet's training and execution times both are faster than AlexNet's. (cf. 2d, Table 3). As was expected, the time for training continues to grow as the number of training examples increases, but this advantages mAlexNet, as the number of examples is bound to increase as we improve out dataset, so the discrepancy between the two models will continue to grow. These facts together point us to employ mAlexNet, instead of AlexNet, from this point on.

LeNet-5 vs. mAlexNet: To this end, learning curves for LeNet-5 and mAlexNet over 10 iterations are possible in a much shorter time window. We demonstrate them in 2c. Compared to 2a, we see the more iterations pay off to smooth out most fluctuations. Not surprisingly, mAlexNet shows comparable behavior and superiority to LeNet-5. This trend is true for both training and validating results over all, but the first few small, samples.

Testing Results

Finally, for we see all three models (LeNet-5 and our two versions of AlexNet) ripe at 90% of training samples, we perform a 10-fold cross validation for them and include the results in Table 4. With this, we observe that AlexNet and LeNet-5 models both produce models that are able to attain 99% accuracy on training and 98% accuracy on validation sets, though AlexNet does it more consistently. Similarly, when comparing the two AlexNet models, the modified version has a slightly higher validation accuracy and training accuracies. Lastly, in terms of testing accuracies, we see that both AlexNet variants are over one percent better than LeNet-5, with mAlexNet a nuance ahead of AlexNet.

Table 4: Results for 10-fold cross validations for LeNet-5 and both AlexNet models. Only the model with the best validation performance is assessed on the test set for each of our three architectures. The top-performing versions of each model have their results in bold.

Model	LeNet-5			AlexNet			mAlexNet		
Val Bucket	Train	Val	Test	Train	Val	Test	Train	Val	Test
1	98.96%	97.46%	—	99.38%	97.67%	—	99.44%	97.67%	—
2	99.04%	98.13%	—	99.40%	98.52%	—	**99.48%**	**98.61%**	**92.41%**
3	99.01%	97.83%	—	99.46%	98.21%	—	99.46%	98.19%	—
4	**99.05%**	**98.41%**	**91.34%**	**99.42%**	**98.54%**	**92.40%**	99.48%	98.54%	—
5	99.07%	96.88%	—	99.39%	97.32%	—	99.48%	98.54%	—
6	99.00%	97.56%	—	99.43%	98.13%	—	99.20%	98.10%	—
7	98.92%	95.97%	—	99.28%	95.93%	—	99.34%	96.10%	—
8	99.08%	96.51%	—	99.49%	97.56%	—	99.46%	97.60%	—
9	98.89%	95.42%	—	99.45%	95.35%	—	99.50%	96.67%	—
10	98.65%	95.95%	—	99.37%	96.15%	—	99.49%	96.82%	—

Conclusion and Future Work

In this paper, we proposed a framework to automate the process of identifying whether an image of marshlands is vegetated, built a new collection of such images, and experimented with CNNs to show their effectiveness. The results we obtained were very encouraging for our overarching multiclass classification problem. With an accuracy of over 92% on a test set for our best model, we have surpassed the expectation of biological scientists and researchers for a model with 80-90% accuracy.

For our learning curves show our models are underfitting, we conjecture that better models could be trained purely through the addition of data. However, because of its limited complexity and lower relative accuracy, it could very well be more prudent to stop working with LeNet-5 and instead focus on improving AlexNet's performance. One such problem is the fact that our test accuracy is comparatively lower than the validation accuracy in our cross validation results. This was not an isolated occurrence either, as it had happened in all three models. This can likely be linked to the lack of representation in the dataset as well as the use of simple random sampling for our buckets for cross validation. These will be remedied in future experiments.

Because many of the classes in our dataset are underrepresented, our first and most important future goal is to expand the size and diversity of our data to more uniformly represent the data. This will done through volunteers at GTMNERR. To ensure the integrity of the dataset, we will administer a 1000 point test to not only these volunteers, but also our other volunteers from the university.

In addition to attempting to refine our binary classifier, we will test several different models on a 5 class species classifier once we have begun to get data from our volunteers, using the expanded dataset on our full problem—determining the species of vegetation present on that point. We will test several different models on species identification—not only our currently tested models LeNet-5 and AlexNet, but also deeper and more complex networks like GoogLeNet and ResNet.

References

[Bacopoulos, Tritinger, and Dix, 2019] Bacopoulos, P.; Tritinger, A. S.; and Dix, N. G. 2019. Sea-level rise impact on salt marsh sustainability and migration for a subtropical estuary: Gtmnerr (guana tolomato matanzas national estuarine research reserve). *Environmental Modeling & Assessment* 24(2):163–184.

[Krizhevsky, Sutskever, and Hinton, 2012] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

[LeCun et al., 1998] LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

[Marcos, Soriano, and Saloma, 2005] Marcos, M. S. A. C.; Soriano, M. N.; and Saloma, C. A. 2005. Classification of coral reef images from underwater video using neural networks. *Optics express* 13(22):8766–8771.

[Mihail et al., 2018] Mihail, R. P.; Cook, W. I.; Griffin, B. M.; Uyeno, T. A.; and Anderson, C. D. 2018. Vegetation density estimation in the wild. In *Proceedings of the ACMSE 2018 Conference*, 9. ACM.

[Tabak et al., 2019] Tabak, M. A.; Norouzzadeh, M. S.; Wolfson, D. W.; Sweeney, S. J.; VerCauteren, K. C.; Snow, N. P.; Halseth, J. M.; Di Salvo, P. A.; Lewis, J. S.; White, M. D.; et al. 2019. Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution* 10(4):585–590.

[Warren and Niering, 1993] Warren, R. S., and Niering, W. A. 1993. Vegetation change on a northeast tidal marsh: Interaction of sea-level rise and marsh accretion. *Ecology* 74(1):96–103.