Holland Computing

Jaelle Kondohoma
Cameron Collingham
Joshua Carini
Kevin Nguyen

05/26/2021

Contents

Abstract

The purpose of this project it to attempt to identify four different types of military aircrafts (fighter, bomber and utility). A Neural Network of transfer learning was created based off of the dataset to train the network to identify if a given picture was a type of military aircraft. Xception is used to classify aircrafts while utilizing transfer learning. A benchmark model was created and utilized on the training set to get results on the unmatured model. The model was applied on the dataset using convolution layers.

Chapter 1

Milestone 1: Project Ideas

1.1 Introduction

We relied heavily on the list provided by Dr Scott. We emphasized the projects that had a proposer: Model-Based Reinforcement Learning (Eleanor Quint), Learning how to apply genetic algorithm operators for software assurance (Hamid Bagheri), and Modifying images with Concepts (Ian Howell).

Additionally, Cameron suggested the problem of identifying military aircraft. After which we scoured the internet for an applicable data set. Luckily, Kaggle had a competition on this very project. So, if we follow through on this idea, we plan to use their data set [8].

1.2 Project Idea 1: Identifying military aircraft by type using Xception

Although other datasets have shown they can identify aircraft by name, if there potentially was a new aircraft in the skies which wasn't named yet it may be difficult to combat it or understand it's intentions. To mediate this we would create a Neural Network of supervised or semi-supervised learning based off of the datasets [8], which already exist, to train the network to identify if a given picture of a military aircraft was either a drone, fighter, or heavy aircraft. an application of this would be early warning systems for tense foreign relations. Another use could be identifying experimental aircraft that have yet to have a designation. There have been previous approaches in solving this problem using objectiveness detection techniques and Xception [1] convolutional neural networks [9].

Table 1.1: Contributions by team member for Milestone 1.

Team Member	Contribution
Cameron Collingham	Project 1 description
Joshua Carini	Intro: Summarize where we looked for project ideas.
Jaelle Kondohoma	Abstract
Kevin Nguyen	Project 2 description

1.3 Project Idea 2: Modifying MNIST Dataset Using Concepts

An image can be visually seen by humans but machines interpret that image differently. There can be small changes made to the image that are not visible to the human eye. We can modify these images using concepts and the machine should be able to interpret the image as something else depending on what concepts are still in the image. There have been cases in which an image has been modified minimally and its whole classification changes. An example of this is Google's AI classifies a turtle to be a firearm [2]. For certain systems, this problem can be very important if humans are utilizing it. Driving a car with an camera detection would need a polished system to detect obstacles or lines on the road. An approach to this problem would be to take an image and modify its concepts to observe how small of changes we can make to the image before its classification changes. For future work ideas, we could possibly go into why these images are seen differently when small modifications are changed and how we can mitigate this type of behavior within systems.

1.4 Conclusions

our results were two options, creating a neural network to distinguish between different type of military aircraft potentially using a supervised learning route or changing items inside MNIST datasets based on characteristics that they may have. For example identifying vertical bars and horizontal bars as concepts in numbers and changing their values by adding or removing concepts. For opinions the first idea we believe it would be easier than project idea 2 but unlike project idea 2 it would lack direct professional advice from Ian Howell. One question our group had is is our Project Idea 2 far enough out of scope to be a good issue. We were thinking instead of MNIST we would use cifar10 instead.

Chapter 2

Milestone 2: Project Selection

2.1 Introduction

We chose to work on "Modifying Images with Concepts" as proposed by Ian Howell. Concepts, with respect to images, are portions of an image that contains a characteristic used to differentiate between images. For example, a concept to differentiate between dogs and humans could be "floppy ears." To see if the neural net is relying on this concept, we could remove the ears from dogs to see if the model still classifies the images correctly. If it doesn't, then we know the neural net is using a "floppy ears" concept to differentiate between dogs and humans. We chose this project because

1. The idea was easy to grasp.
2. It emphasizes understanding why a model classifies inputs the way it does, which is important in improving a model and/or correcting errors in a model.
3. Ian has been very approachable and has done a great job explaining some of the details, and he will continue to advise us as the project moves forward.

2.2 Problem Specification

Humans are able to distinguish concepts in forms of characteristics or objects/entities that describe a certain scenario. The problem is that machines recognize concepts differently than humans and may classify images incorrectly. Their perception of images are more intricate than just colors and shapes. Small changes in certain features or parameters of an image can yield false labels, which are adversarial images. Adversarial attacks have been used by hackers

to exploit criterion of deep learning models, which can fool AI security and so is a big risk to these systems [10]. For the scope of our project in relation to the problem, certain images with minor concept contrasts may change the whole classification of the image. Trained classifiers are efficient in identifying concepts, in which we will utilize in our project to observe our model's behavior when image concepts are modified. This is based off of Ian Howell's work [4].

2.2.1 Motivation

Through the evolution of technology, classification problems are commonly researched to provide for new features in software. Machine vision can be popularly known as facial recognition in phones [6] and used in self driving cars. Image perception is important for these systems and our project goal aims to dive into how these systems may label modified images in contrast to the original image.

2.2.2 Resources

Some of the resources we will be using are:

1. NumPy
2. TF-GAN
3. Matplotlib
4. TensorFlow
5. MNIST dataset

2.3 Proposed Method 1: Image Segmentation Replacement

For a first step, we must train the Convolutional Neural Network which will be our classifier on MNIST or, if possible, find one that has already been made. This is used to classify the different concepts that create a digit in an image. Next, we will make use of the generative model we have selected which is a deep convolutional generative adversarial network over the MNIST dataset. After that we must train a concept-recognizer to to identify the components. For example, a 9 is made out of a variety of horizontal lines and vertical lines. These instructions are based off of Ian Howell's work [4].The machine will be trained to replace a concept from a position and put it into the place of another. To evaluate the performance of this we will use the tf.gan library function Kernel Distance to compare our modified digit to another. A definable midpoint of this would be ensuring that the network is at least capable of identifying the concepts of a digit but the final milestone would be it rearranging the concepts to make and identify a new digit comparing it as stated above.

2.4 Proposed Method 2: Image Segmentation Deletion

Our end goal is to optimize an image to present or not present a concept. Our pipeline for this method will be to first train the a classifier on a Convolutional Neural Network. The classifier will be used to identify different concepts used in order to create a digit in an image. We will be using an MNIST pre-trained generative model. A concept organizer will be used to identify concepts. For this second method a section of an image will be removed and replaced with a background then we will attempt to see if we can still predict if its the same number. Our performance will be evaluated using the 'gan' (generative adversarial network) function in the TensorFlow library.

2.5 Conclusions

To conclude, image classification is a widespread topic in deep learning. The problem of image classification is that since machines perceive images differently than humans do, they are prone to miscalculated errors when trying to assign labels to a picture. For our project, we decided to change existing concepts within an image to observe how its labeled and its performance. Our proposed ways of modifying the MNIST dataset involves identifying the concepts of a digit and modifying/replacing them to create a different digit or symbol. Both of our methods are similar in the way they will be made, trained, and tested but we believe the second proposed method would be best suit for our abilities as it takes another moving part out of the equation.

Table 2.1: Contributions by team member for Milestone 2.

Team Member	**Contribution**
Josh	Introduction and updated abstract
Cameron Collingham	Proposed method 1
Kevin Nguyen	Problem Specification
Jaelle kondohoma	Proposed method 2

Chapter 3

Milestone 3: Progress Report

3.1 Introduction

In this Milestone we began working with the issue of transfer learning off of the dataset provided by kaggle to begin using Xception to classify aircrafts. Using an initial benchmark model, we can obtain introductory results that would allow us to change some layers our model and re-purpose our second task using all the resources and data gained from our first task (using the same model). This saves us a lot time and effort since we will not need to go through the process of training a neural network from scratch while fine tuned to our dataset. So far the dataset has given us the ability to start benchmarking using a simple convolution.

3.2 Experimental Setup

Our setup for our first experiment was to set a benchmark for the error so we can compare that with the error we would get otherwise. We first set up our project by installing kaggle and uploading a specific file called "kaggle.json". In order to download our dataset without locally importing in the files every session, "kaggle.json" is a required token to authorize our kaggle accounts. After downloading the image dataset, we load a single image from the dataset and we use Xception [1] to classify the image and observe our results. This is done to verify that our images would even be able to be classified using Xception to begin with. The next step in setting up our project is assigning our data and labels to lists so that we may begin splitting our dataset into training, validation, and testing sets. We iterate through our training set and shuffle the minibatches. The purpose of this step is to set up and create a benchmark model that would allow us to analyze our initial results and accuracy when we

train that model for several epochs using our data. We expect our results to not be too accurate at the beginning. We can use these results as a reference point when we continue on to utilize transfer learning in our project. For the transfer learning step, we will reuse our benchmark model as the starting point of our second task, which will be to fine tune our model for better and more accurate results.

3.3 Experimental Results

Our experiment results unfortunately were hampered by multiple elements. Mainly, the dataset which we were given was not easy to form into usable data and created many errors in the training function. However the following graph should accurately project our estimated values

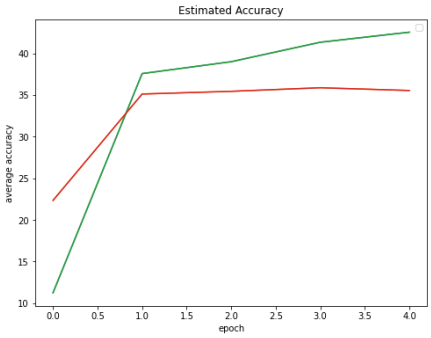


Figure 3.1: Graph representing Accuracy. Green is for training, red is for validation

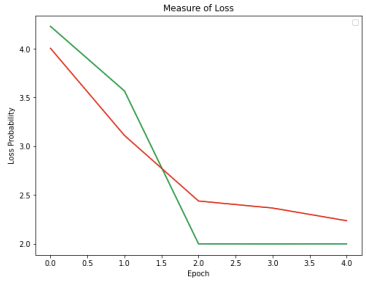


Figure 3.2: Graph representing Measured Loss. Green is for training, red is for validation

3.4 Discussion

The results of our experiment were as follows. Our experiment would not have a very good accuracy value given the allotted epochs used during our benchmark, as the benchmark was only suppose to provide a general idea of where our accuracy's and losses would fall. Part of this is due to the rather small size of the dataset [9]. An excellent solution to this issues would be using an image generator such as ImageDataGenerator provided by keras's image preprocessing suite. The larger we can inflate our dataset the more accurate our results will be. For further experiments we will increase the training epochs as well. The benchmark testing may have been good for a start but further testing should yield better accuracy and loss values.

3.5 Conclusion

In conclusion, beginning with a benchmark model provides an excellent foundation for us to utilize transfer learning to classify air crafts. Since we are using a rather small dataset, the process of optimization is more difficult so transfer learning would be a great solution for this problem. Starting with a pre-trained initial model, we can make sure general features of the image are identified before moving on to more complex and specific features. Since Xception is already pre-trained with many images, it can be useful for us to use their predefined features to supplement us in classifying our air craft dataset.

Table 3.1: Contributions by team member for Milestone 3.

Team Member	**Contribution**
Cameron Collingham	Exp. Results, discussion, python code
Kevin Nguyen	Experimental Setup, research, conclusion
Jaelle Kondohoma	Abstract, research
Joshua Carini	Code, data cleaning

Chapter 4

Milestone 4: Final Report

4.1 Introduction

In this milestone we began working on transfer learning. Transfer learning will be vital to our project's results since we have a small dataset, so we will be augmenting our data and modifying the layers of our transfer learning model. Now that our benchmark is completed we will be using the weights from the pretrained model Xception [1] to compare to the model from "Image classification from Scratch" [3]. Since Xception specializes in image classification, we expect to achieve better accuracy and lower loss, which would show that training a well established model is more efficient and more accurate then creating a whole new model. Thus far, this hypothesis has been correct as the accuracy in the Xception model rose up a fair amount in comparison to the benchmark. We will present exact values in the next section.

4.2 Experimental Setup

Data Cleaning: We used the Military Aircraft dataset on Kaggle. [9] We separated the files by their class name (e.g. F22, A10), and then copied the images within each class into bomber, utility, and fighter folders according to our local expert, Cameron. These classes were then partitioned into train, validation, and test. We also had a 'drones' class, but removed that class and the images it contained since there weren't many drone images.

Pre-processing: We checked our dataset for corrupted files, and set the shape of our images to [150, 150, 3]. Next, we inflated our aircraft dataset from approximately 2800 images to a total of roughly 20,000 by way of random flip, which was set to "horizontal", and random rotation, which was set to (0.1) (as well as a combination of the two).

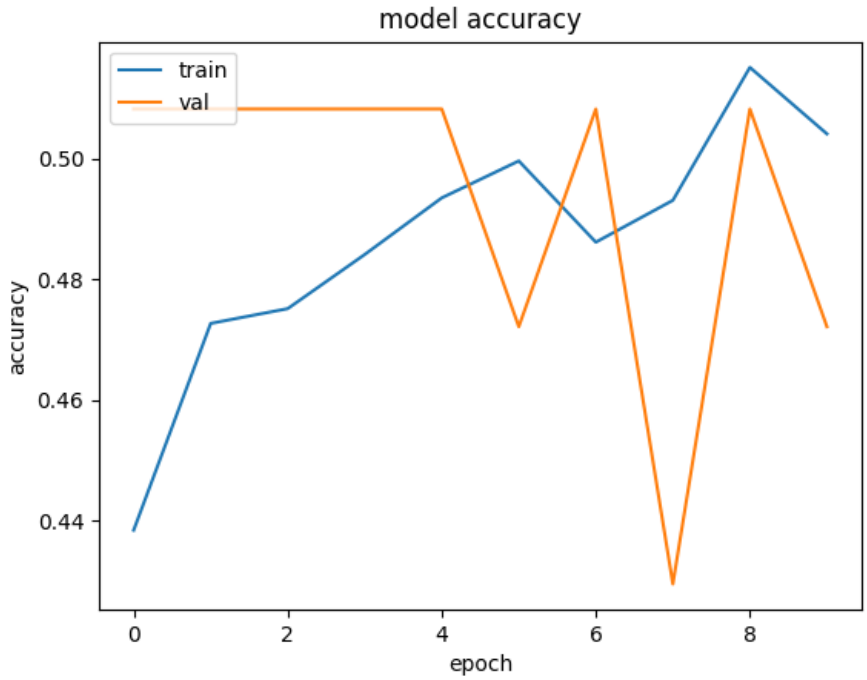
Model training: We loaded the Xception model, froze the weights and substituted our layers for the Xception output layer. We compiled using stochastic gradient descent (it seemed to work better than Adam) with the learning rate set to .01, and trained the model for 10 epochs. We then fine-tuned this model by unfreezing the Xception layers except for the BatchNormalization layers (with training=False), set the learning rate to 1e-5, and trained the model for an addition 10 epochs.

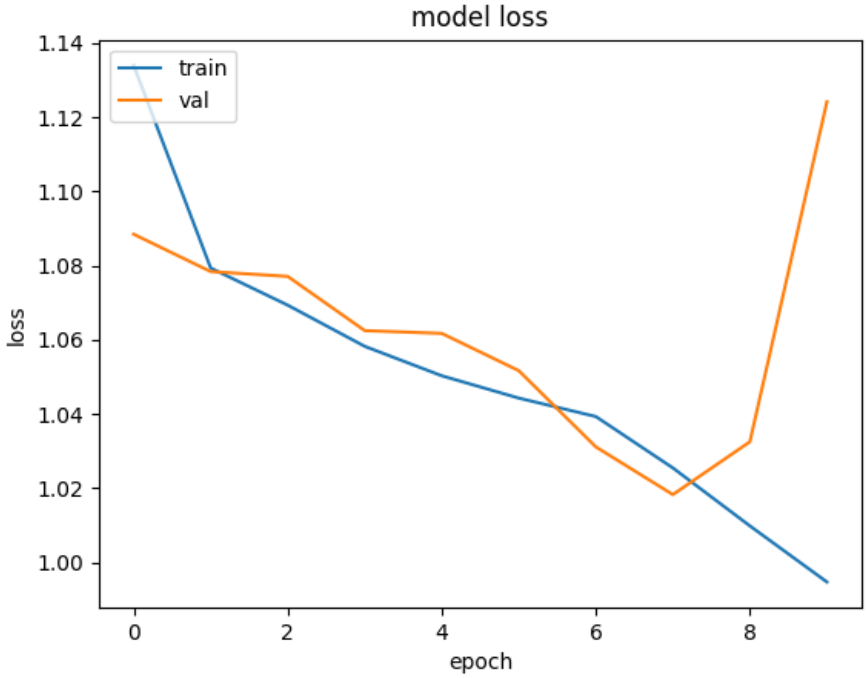
Evaluation: We kept tract of loss and accuracy with metrics = "accuracy" for each epoch, and used model.evaluate to get accuracy on our test set.

4.3 Experimental Results

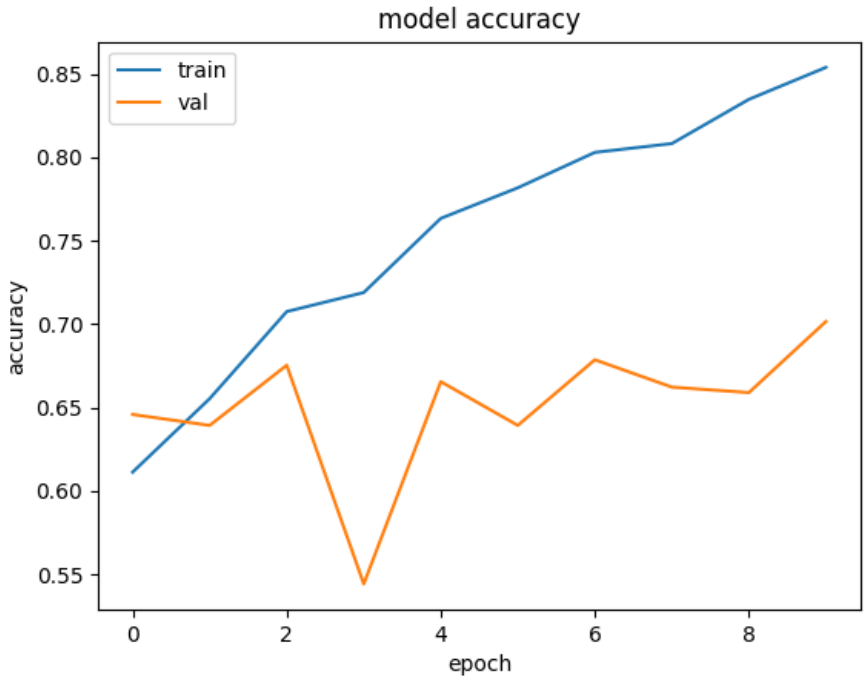
The result we gathered were almost exactly what we expected. During our experience transfer learning we managed to gain a good amount of accuracy as now we achieve up to 85% on recent tests. Loss has also gone down due to higher epoch counts and a more robust dataset. Our benchmark model had varying validation accuracy values that are visible in the graph. The loss value near the tenth epoch was a clear outlier. For our transfer learning model, the accuracy of the training set increased at a stable level when compared to the validation set. The transfer learning loss graph shows a significant difference of the loss values between the training and validation sets.

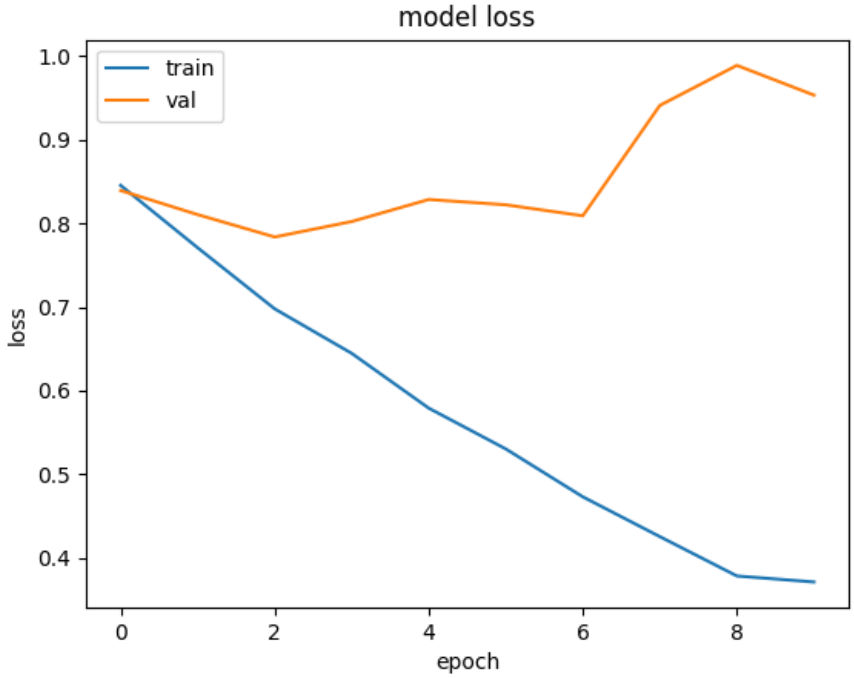
4.3.1 Benchmark Accuracy and Loss





4.3.2 Transfer Learning Model Accuracy and Loss





4.4 Discussion

When comparing our benchmark model to our transfer learning model, it was clear that the transfer learning model performed better than the benchmark model. By analyzing the graph, we can see that the accuracy of the training set gradually had a higher peak than the benchmark model. The benchmark model had various dips from what we can see. Within ten epochs, our transfer learning model was better. However, we can see that in the transfer learning model graphs the validation accuracy and loss diverged away from the training accuracy and loss. This is a clear sign that our model may be overfitted. This is a problem we initially had in mind since we knew our project would be prone to overfitting due to our small dataset size [5]. We had initially tried to solve this problem in the beginning by augmenting our data in the training dataset in order to have more examples to work off of. The fact that the model is still overfitted means that we may have to implement early stopping in our network or adjust the dropout layer and decrease the network size. That may potentially fix our loss values and provide a better fit for our model. Due to the time constraint and behavior of the validation accuracies, we decided to run our model for ten epochs. If we had ran it for more epochs, we would likely see the graph lines become settled. In the end however, without fixing the underlying cause for our overfitted model, it would still be difficult to yield acceptable results. There were examples in our dataset where images would include aircrafts in newspapers or reflections off surfaces. Our model would not do well to classify these images if cases like these images were to appear. Another factor that may have led to our overfitted model could be due to our network being over-parameterized for our training samples [7].

4.5 Conclusion

In conclusion, once we decided on selecting three military aircraft classes, we split up our dataset into training, validation, and test to begin with. We were able to create a benchmark model that would help serve as an initial point for us to compare results from our transfer learning model. Using this benchmark model, we were able to visualize its accuracy and loss values on a graph to see how the model was fitted using our datasets. Our expectation for this model was quite low, and we expected our transfer learning model to perform better and fit our dataset once we change layer parameters. Our results revealed to us that although transfer learning is optimal for our project (due to the small amount of images in our dataset), the model is overfitted. Down the line, this will eventually lead to poor performance on unseen data.

Table 4.1: Contributions by team member for Milestone 5.

Team Member	**Contribution**
Cameron Collingham	Code, Report
Josh Carini	Experimental Setup, Code, Data cleaning
Jaelle Kondohoma	Presentation
Kevin Nguyen	Code, Presentation, Report

Bibliography

[1] ? Xception. accessed: 2021-07-08. URL: https://keras.io/api/applications/xception/.

[2] A. Athalye, L. Engstrom, A. Ilyas, K. Kwok. Synthesizing robust adversarial examples. Technical report, Massachusetts Institute of Technology, Cambridge, MA, 2018.

[3] fchollet. Image classification from scratch. accessed: 2021-07-08. URL: https://keras.io/examples/vision/image_classification_from_scratch/.

[4] Ian Howell. Concept optimization with xgems. Technical report, University of Nebraska - Lincoln, Lincoln, NE, 2020.

[5] Hong-Wei Ng, Viet Dung Nguyen, Vassilios Vonikakis, and Stefan Winkler. Deep learning for emotion recognition on small datasets using transfer learning. *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, 2015. https://doi.org/10.1145/2818346.2830593 doi:10.1145/2818346.2830593.

[6] Ríos-Sánchez, Belén and Costa-Da-Silva, David and Martín-Yuste, Natalia and Sánchez-Ávila, Carmen. Deep learning for facial recognition on single sample per person scenarios with varied capturing conditions. *Applied Sciences*, 9(24):5474, 2019. https://doi.org/10.3390/app9245474 doi:10.3390/app9245474.

[7] Shaeke Salman and Xiuwen Liu. Overfitting mechanism and avoidance in deep neural networks. 2019. https://doi.org/10.1145/2818346.2830593 doi:10.1145/2818346.2830593.

[8] T. Nakamura. Military aircraft detection dataset. accessed: 2021-05-28. URL: https://www.kaggle.com/a2015003713/militaryaircraftdetectiondataset/code.

[9] Wu, Hui and Zhang, Hui and Zhang, Jinfang and Xu, Fanjiang. Fast aircraft detection in satellite images based on convolutional neural networks. In *2015 IEEE International Conference on Image Processing (ICIP)*,

pages 4210–4214, 2015. https://doi.org/10.1109/ICIP.2015.7351599 doi:10.1109/ICIP.2015.7351599.

[10] Zhang, Shaofeng and Wang, Zheng and Xu, Xing and Guan, Xiang and Yang, Yang. Fooled by imagination: Adversarial attack to image captioning via perturbation in complex domain. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2020. https://doi.org/10.1109/ICME46284.2020.9102842 doi:10.1109/ICME46284.2020.9102842.