Spring 5-31-2021

Machine Learning for Terminal Procedure Chart Change Detection

Anthony M. Marchiafava
University of New Orleans, amarchia@uno.edu

Machine Learning for Terminal Procedure Chart Change Detection

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

by

Anthony Marchiafava

B.A. Louisiana State University, 2014

May, 2021

Table of Contents

List of Figures

List of Tables

Abstract

Terminal Procedure Charts are a constantly updated and necessary tool for aircraft personnel to approach and take off from airport runways safely. Detecting changes within these charts is a time-consuming and laborious process. Here machine learning techniques were used to predict regions of change in charts based on detecting the charts image regions and comparing features extracted from those regions. Outlined are methodologies to detect differences between two separate charts to produce images with changed regions clearly indicated. Both more conventional computer vision and machine learning techniques were applied. For images with minor shifts, the proposed model is able to ignore them to a greater degree than the baseline.

Keywords: Machine Learning, Computer Vision, Change Detection

Chapter 1 – Introduction

The topic of computer vision is an important one in the broader scheme of computer science. Often the purpose of a computer system is to augment or assist a human in achieving some highly repetitive and monotonous task. One such task is the detection of changes between similar images. These change spotting techniques have been used in such mundane settings as the sort of games for children one might find in a newspaper, to using "blink" comparing techniques used to detect changes in images from telescopes. Computer automation has allowed for much more advanced comparison techniques.

One task of note would be the comparison of new and old aeronautical navigation aids and charts. These charts are published in a quick cycle, and manually analyzing them is a time-consuming process that leaves room for human error on the part of the observer. An automated system capable of rapidly examining two charts and indicating to the human observer where the two charts differ would be a powerful way of augmenting and accelerating the rate at which human observers could examine these charts and track their changes.

Chapter 2 – Literature Review

To begin the topic of computer vision, first consider the topic of human vision. We can imagine the human eye as a spherical camera with a focal lens outside, focusing an image on the retina opposite to the lens with roughly one hundred million receptor cells [1]. Some receptors take in color while others take in black and white signals where each signal is the intensity of the wavelength of the light received at that point. The brain is then responsible for processing the results of these visual inputs for human use.

Computer representation of images is somewhat related in that computers store images in matrices whose rows and columns store pixels. These pixels store a fixed range of numbers representing the brightness at that given location in the image. Therefore, an image made up of a gradient from black to white pixels would be expressed as a grayscale image containing pixels whose values range from 0 to 255. For the purposes here, the image data's reception is less significant than the image data's storing and analysis as pixel values. These values are what the computer system has available to use. The goal then is to use the pixel data values to analyze an image's contents such that the computer can make decisions based on image contents. Here specifically, the computer will need to analyze two images and discover where they are dissimilar using different techniques derived from these pixel values.

2.1 Pixel to Pixel Comparison

The techniques traditionally used for image change detection can be applied to pixel-level, feature-level, or on an object-level. They can be classified into groups such as manual visual analysis, generating a change map through algebraic operations, reducing data through transformations, or using classification-based methods [2]. The manual analysis of two images is a method by which a trained human analyst examines two images and notes what has changed. Algebraic operations, transformations, and classification methods attempt to approximate the result of manual verification using automated computer tools and algorithms. These methods are used to do change analysis when detecting differences between two images and detecting changes in series of images over time. The specifics of what is a significant change may be domain-specific,

such as ignoring background change information in surveillance footage and focusing on background information in satellite imagery and remote sensing [3].

In each case, the goal is to detect the changes an otherwise human observer would see and find meaningful, while the observer should ignore sources of noise and otherwise irrelevant changes. Direct pixel to pixel comparison is a straightforward and relatively simple place to start detecting changes. To do so, one matches up the edges of both images' scales both images to some uniform size, and subtract one image data from the other image data to show what has changed. This algorithm's simple form would be to have some threshold value and create a mask where the difference in the two images is shown for values above that threshold value. The mask would indicate the notable regions of change. A technique like this would be useful for examining two images, which are taken of a scene from the same perspective at two different points in time. A major downside would be that this technique is very sensitive to noise and variations of illumination [3]. Something like mean square error (MSE) could be used to define the difference between two images. Take one of the images to be compared as the image should be and use the other image as the potentially altered version of the original image. Each pixel subtracts the values of the altered version from the original. Then square each difference and average those distances together.

2.2 Structural Similarity Index Measure (SSIM)

One more complex methodology used in signal analysis for detecting image loss would be a structural similarity score. Used in various fields for quantifying change in image signal quality, this metric works by comparing one potentially distorted image to some ground truth image. The Structural Similarity Index Measure (SSIM) is used as a metric for image quality instead of a metric like MSE because it emphasizes the visually perceived changes. It does this by comparing the luminance, contrast, and signal values in greyscale versions of the two images to be compared [4]. SSIM can be used in the case of change detection for comparing two similar images to detect when things have changed and given a quantitative measure of the extent of that change [5].

2.3 Keypoint Detectors

Image features are also detectable through conventional algorithmic methods. These techniques use methods of detecting features and assigning feature descriptions to them. One common algorithm used for detecting and describing important points is ORB (oriented FAST rotated BRIEF). It is scale, rotation, and affine invariant. The features detected are often in the form of corners, blobs, edges, junctions, and lines, and these features are described based on patterns of nearby neighboring pixels [6]. These generated key points are used in tasks like object detection or image stitching. It detects FAST points in an image and uses a Harris corner measure to order the key points to find the most relevant FAST points [7]. FAST points are used for detecting key points via finding corners in the image [8]. The Harris corner measures also detect edges and measures the edge quality [9]. ORB then uses the BRIEF key point descriptor with corrections to allow it to be rotation invariant by finding the orientation of the key points and use a pre-computed table of BRIEF patterns to select the correct set of points for the descriptor. These BRIEF patterns are shorter key point descriptors in binary strings computed from image patches [10].

These descriptors can be used to compare images by the Hamming distance between the two strings, which is the number of positions where the two strings are different. One example of ORB's detection may be to take two photographs with some overlapping central feature and find similar points between the two images to find the manipulations needed to align the two images and stitch them together with a panorama. The detected features could then be used to compare the two images and note which features are present in one and not the other. However, this could lead to issues where a similar feature is erroneously detected somewhere else in the image or the feature is shifted a considerable amount. Instead of comparing pixel to pixel regions, a mechanism, which would be even more capable of accounting for greater shifts in pixel locations, would be preferable.

2.4 Machine Learning Techniques

Aside from the previous keypoint detection mechanisms, there are other ways of extracting features derived from examples of images. Image classification is a task that would require discovering features within images using previously labeled images so the relevant features could

be discovered. The approach of using feedback from input-output pairs for a program to learn some function to map an input, here an image, to the output, here the classification of the object within the image, is known as supervised learning [11].

2.4.1 Simple Linear Regression

A simple example of this supervised learning technique would be linear regression and classification. This simple example has one input variable *x*, one output variable *y*, and is based on a straight line. This approach's goal would be to find the line that best fits a series of input values based on applying weights to the input. Here we could use the equation of a line such that the output *y* is equal to *m* multiplied by *x*, and some *b* is then added, where *m* and *b* are coefficients (or weights in this context). When inputs and the corresponding outputs are already known, a loss function can be used to find out the error produced by any set of coefficients. One prominent example of such a loss function would be the residual sum of squares (RSS) [12]. This finds the error of a given function by summing the square of the difference of what the value actually is and the value that is predicted by some function f(x) squared (see Equation (1)).

$$RSS(f(x)) = \sum_{i=1}^{N} (y_i - f(x_i))^2 \quad (1)$$

Ideally, this allows the estimation of those weights by adjusting some arbitrarily chosen weights by increasingly minimizing that loss [12]. Finding the weights that create the line that best fits the data is known as linear regression. This is done by finding the partial derivatives of the loss function with respect to the two weights to find where each would be equal to zero and use those to find what those weights should be. This problem can be generalized to be an optimization problem solvable by a hill-climbing algorithm that follows the gradient of the function to be optimized, and since it is attempting to minimize the loss, this is known as gradient descent [11].

This can be generalized to linear regression for problems with many variables to find the best vector of weights, which are applied to a vector of input. One notable and efficient method of using a machine learning technique derived from linear regression to detect change detection features would be deep artificial neural networks [13] [14].

2.4.2 Artificial Neural Network

An artificial neural network is a network of neurons inspired and in part modeled by the connections of brain cells in the brain. The neuron activates when a linear combination of its inputs exceeds some hard or soft threshold expressed as a function. This allows for a linear classification mechanism, which delineates between two potential classifications, either activated or inactivated. The neural network is made up of many of these activation units connected together, and the network itself has properties determined by the structure and order of these connections and the properties of these neuron-like nodes [11]. The connections are made up of directed links connected from node *i* to node *j*, which propagates the activation a_i from *i* to *j*. These links also have numeric weight ($w_{i,j}$) associated with them, which ties into the direction's strength and sign. There is also a dummy input $a_0=1$ with weight $w_{0,j,}$ and each node j computes a weighted sum of its inputs then applies an activation function to get the output (see Equations (2.1, 2.2)) [11].

$$in_j = \sum_{i=0}^{n} w_{i,j} a_i \tag{2.1}$$

$$a_j = g(\sum_{i=0}^{n} w_{i,j} a_i) = g(in_j) \tag{2.2}$$

When these nodes are arranged in layers where each unit receives input only from units in the preceding layer, this is usually known as a feedforward network. The simplest example of these would be a single layer where each of the inputs is directly connected to the outputs, known as a perceptron network. These perceptron networks can learn through a process similar to linear regression or logistic regression. The general idea being for data that is linearly separable, there is some equation such that for given inputs, if a value is above that equation, one should classify it as one type and below that equation, not of that type. The activation functions, as previously noted, would be the sort of threshold function that would allow the result to be 1 if the threshold is met and 0 if otherwise. The weights of these activation functions could be updated in a manner like the previously stated linear regression technique.

Multilayered feedforward neural networks are also possible. These networks have at least one and potentially more than one hidden layer in between the initial input and the output. When

multiple hidden layers are used, these networks are also known as deep neural networks. The error must then be calculated, and the hidden layer weights updated because the training data do not say anything about what values the hidden nodes should have. This problem is solved through a technique known as backpropagation. Each hidden node's error is responsible for a fraction of each output node's error to which it connects to. Essentially, gradients can be computed by efficiently propagating from the output to the input [15].

2.4.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a special type of deep neural network used to extract features within images and have been used to effectively classify the contents of images based on these features [16] [17]. These neural networks are trainable and make strong predictions about an image, and are created from training multistage architectures made up of multiple stages where the input and output of each stage is a set of arrays called a feature map. With the example of two-dimensional color images as the initial input, each feature map would be a two-dimensional array of color channels of the image, and the output of each feature map would represent the features extracted at all locations on the input. In its simplest form, a convolutional network is made up of a filter bank layer, a non-linearity layer, and a feature pooling layer where the previous output is fed in as input [18].

The point of most interest here is the filter bank layer where the input would be some three-dimensional array made up of feature maps denoted as $x_{i,}$ which are made up of components $x_{ijk,}$ and the output would be a different set of feature maps based on the initial input y_j. To get from each input feature map to each output feature map requires the use of a trainable filter (kernel) k_{ij} in a bank of filters which connect x_i to y_j where * is the convolutional operator b_j is a trainable bias parameter (see Equation 3).

$$y_j = b_j + \sum_i k_{ij} * x_i \tag{3}$$

This allows for each filter to detect a particular feature at every location on the input. These filters are used in a similar fashion as when applying a mask to an image. The mask would be a set of pixel positions and weights, where the application of the mask onto an input image would conceptually place the origin of the mask over each, and every pixel (if the stride of the mask is one, or each other pixel if the stride was two) of the input image and the value of each pixel under the mask is multiplied by the weight of the mask pixel [1]. The results would be summed together

and placed in a smaller (based on the size of the mask) matrix. In some examples, to keep the same spatial dimensions adding virtual rows may be needed, and zero paddings may be used to compensate for the lost values.

The trainable filter (k_{ij}) in the bank of filters would behave similarly to the previously mentioned masks, but instead of storing pixel values, the output would be how close a feature matches a given filter in that location. These outputs are then fed to a non-linearity layer which would use some activation function applied to each output of the convolutional layer, such as the tanh() sigmoid function or the rectified linear activation function to get the activation map. This allows the modeling of the neuron's output as a function of its input in a non-linear fashion [17], and this is advantageous for modeling decisions, which do not map directly to linear functions [12]. A layer using the rectified linear activation function, when used as a rectified linear unit (ReLU), is especially common partially due to its proposed intensity invariance and non-linearity [19].

The result of the non-linearity layer is then fed to the feature pooling layer, which looks at each feature map (without consideration to each other feature map) and computes average values or the max values over a neighborhood in that feature map. This allows for a reduction in the feature map's resolution as multiple values can be replaced by the pooled value. Using a stride of two, for example, would reduce the dimensions of the feature map further in a similar fashion to how the mask behaved. Alternatively, one could use a stride, on the feature map, of greater than one to make a similar reduction in the filter bank layer. CNNs allow some degree of shift, scale, and distortion invariance as connecting layers of these stages together allow visual features such as oriented edges, endpoints, and corners to be combined by a subsequent layer to begin to detect higher-order features [15]. These higher-order features can then be used by subsequent fully connected (non-convolutional) layers to predict potential classifications.

2.4.4 Region-Based Convolutional Neural Network

CNNs may be useful for classifying an image's content but to detect where a classifiable object is in an image maybe requires further analysis. This is where region-based convolutional neural networks (R-CNN) are useful. Implementations of R-CNNs must account for multiple objects of different aspect ratios and image sizes. These can use supervised learning on a large pre-training auxiliary dataset followed up with fine-tuning using a smaller domain-specific dataset.

The approach takes an input image, extracts a number of bottom-up region proposals, compute features for each proposal using a convolutional neural network, and classifies each region using a class-specific support vector machine [20]. There are multiple developments on top of the initial R-CNN to account for this issue's speed, such as Fast R-CNN, which took regions of interest and turned them into the region of interest pools [21]. Each region of interest uses max pooling to convert the features into a small feature map. Faster R-CNN follows up on this by adding that proposals can be computed with a deep convolutional neural network called a Region Proposal Network (RPN) [22]. This RPN regresses region bounds and objectness scores at each location onto a grid, and as a fully convolutional network, this can be trained end to end for creating detection proposals. This means that for a given image, Faster R-CNN generates feature maps of the original image, proposals from the region proposal network and uses the contents of the features matched with the region proposals to detect and classify objects within the/a scene.

2.4.5 Support Vector Machines

Within versions of R-CNN, there must be some way to take the features and classify them. The original R-CNN network would take these proposals of regions with likely classifiable objects and create a feature vector for that region, then pass this feature vector to a support vector machine (SVM). What these do is find some line of optimal separation between two datasets. It does this by transforming the data into extra dimensions so that points could be linearly separable in higher dimensions where they may not be separable in the dimensions they are in. Instead, this is done by transforming the data that generates new features derived from the already existing features in the data [23]. Originally, these were used for classification purposes. However, the R-CNN model replaces the use of an SVM to classify each potential object with the use of a softmax layer. This softmax layer instead reports the probability of a given element belonging to a particular class. These are reported to get performance comparable, if not sometimes better, to an SVM in classification predictions [20].

2.4.6 Mask R-CNN

Mask-R-CNN is a further development following from Faster R-CNN. This approach is designed for image object detection and segmentation, where the goal is to classify each pixel into

a fixed set of categories. This is done by extending Faster R-CNN with an additional branch for predicting segmentation masks on each region of interest in parallel with the existing branch for classification and bounding box regression. This allows the use of the region of interest alignment to faithfully preserve exact locations for pixel-to-pixel alignment between network inputs and output [24]. This is done to increase mask accuracy. The end result is a neural network that can detect objects within scenes and can segment individual instances of those objects even if those objects may be nearby or overlap.

2.4.7 Siamese Neural Network

One other major consideration and topic worth discussing is the idea of a Siamese neural network. These are designed to take two images as input. Siamese networks have been used successfully for change detection in the field of remote sensing change detection of satellite images with both multispectral images and also red, green, and blue (RGB) color images [25] [26]. These Siamese networks would take the input images in parallel, pass them separately through parallel convolutional networks with shared weights, and create output from concatenating the outputs [27]. This would require substantial amounts of ground truth data where the pixel to pixel comparisons would otherwise be roughly the same location and has been used on multispectral images with more than just RGB image inputs. This approach would allow inputs, which have more spectral data than just RGB data, and could include that data in the comparison analysis. Entirely new and trained from scratch networks may be needed when handling these multispectral inputs. The types of charts that will be compared here are merely made up of RGB images. However, inspired by these Siamese networks, an approach using the features generated by a network and comparing them to features generated from the same network on another image, at different convolutional layers, would be a way of detecting changes from one chart to another that would not require extensively trained change region annotations.

Chapter 3 – Experimental Materials

The goal of this work is to improve the analysis of Aeronautical Charts found within Terminal Procedures Publications. These charts include Instrument Approach Procedures (IAP)

Charts, Airport Diagrams, Departure Procedures (DP), Standard Terminal Arrival (STAR) Charts, and Charted Visual Flight Procedure (CVFP) Charts.

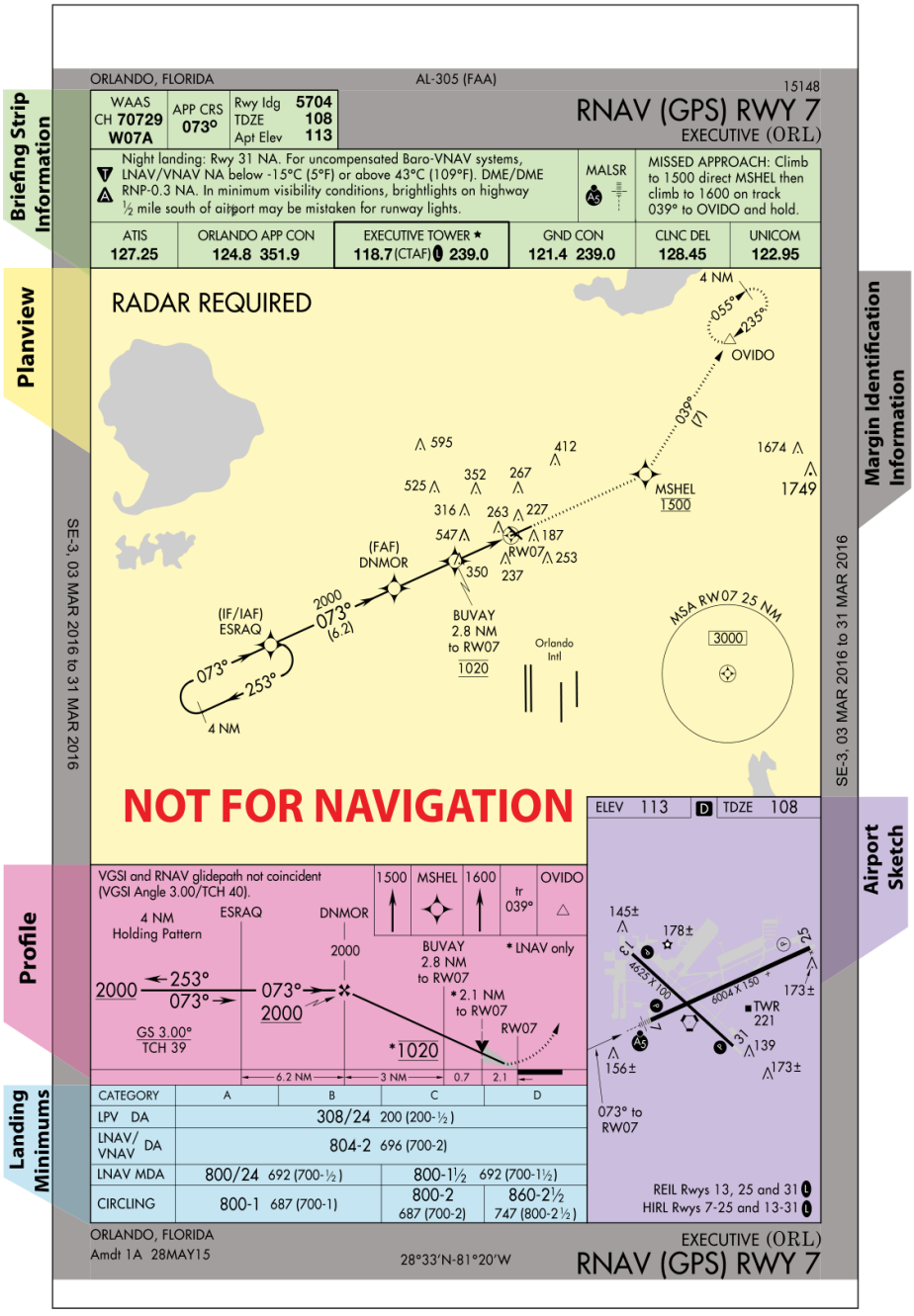

Figure 1: Instrument Approach Procedures (IAP) example

Any IAP, as an example see Figure 1, is divided into sections. One section includes identifying information such as the location and airport name information. Another includes a briefing strip about navigation, approach information, and runway landing data. There should be

a sketch of the airport with details on runway patterns and information for identifying the runway from the air. There should also be missed approach information with a profile view to show descent profiles, various settings, and landings and values needed to help navigate. There is also a graphic called a planview. This planview includes a graphical representation of the entry procedure through a missed approach accompanied by a map with important features of the nearby area to help show visual flight paths, navigation aids, and holding pattern procedures [28].

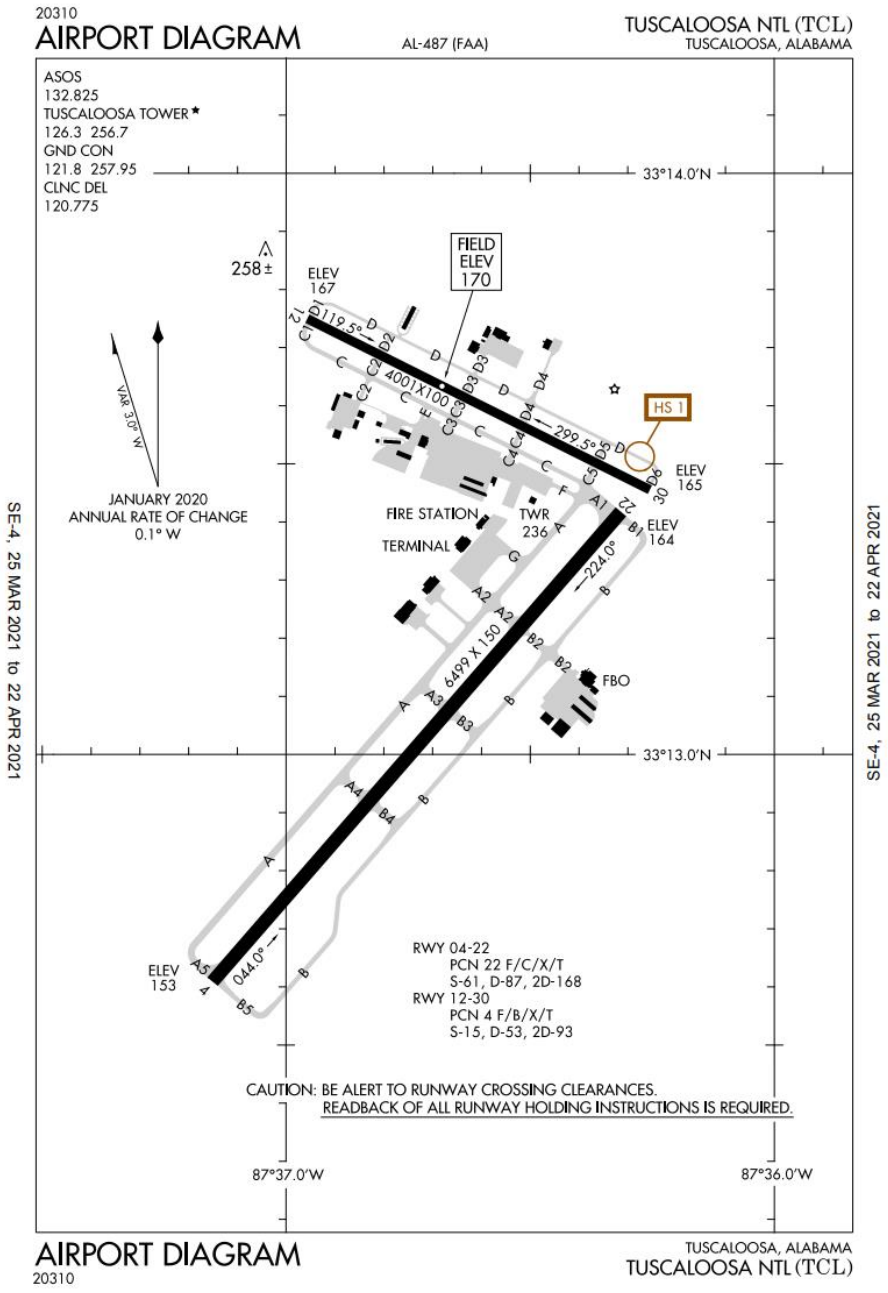

Figure 2: Airport Diagram Example

Airport Diagrams are more complex visual representations of airports, emphasizing runways, boundaries, radar reflectors, and other obstacles in addition to communication frequencies.

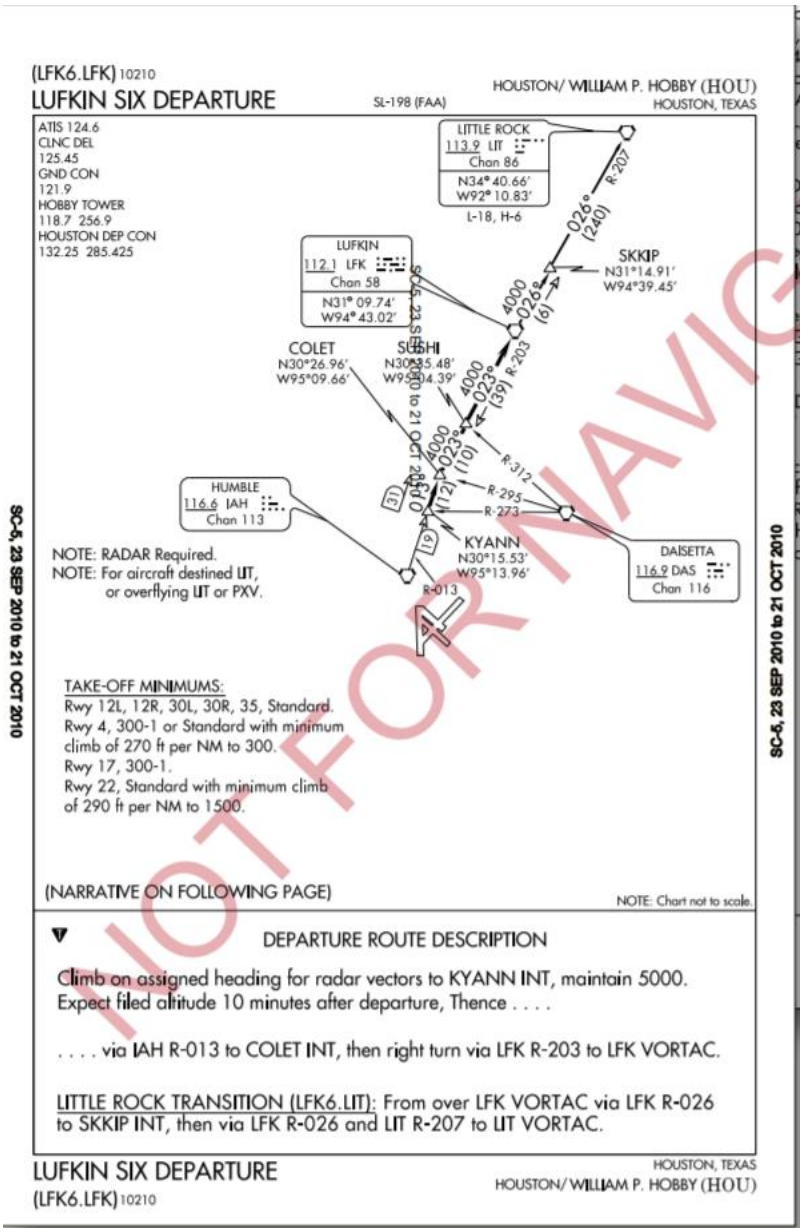

Figure 3: SID Example

Departure Procedures are designed to assist pilots in avoiding obstacles on takeoff and are either Obstacle Departure Procedures (ODPs), which are sometimes graphical, or Standard Instrument Departures (SIDs), which are always graphical. DPs are generally not printed to scale due to the large distances involved.

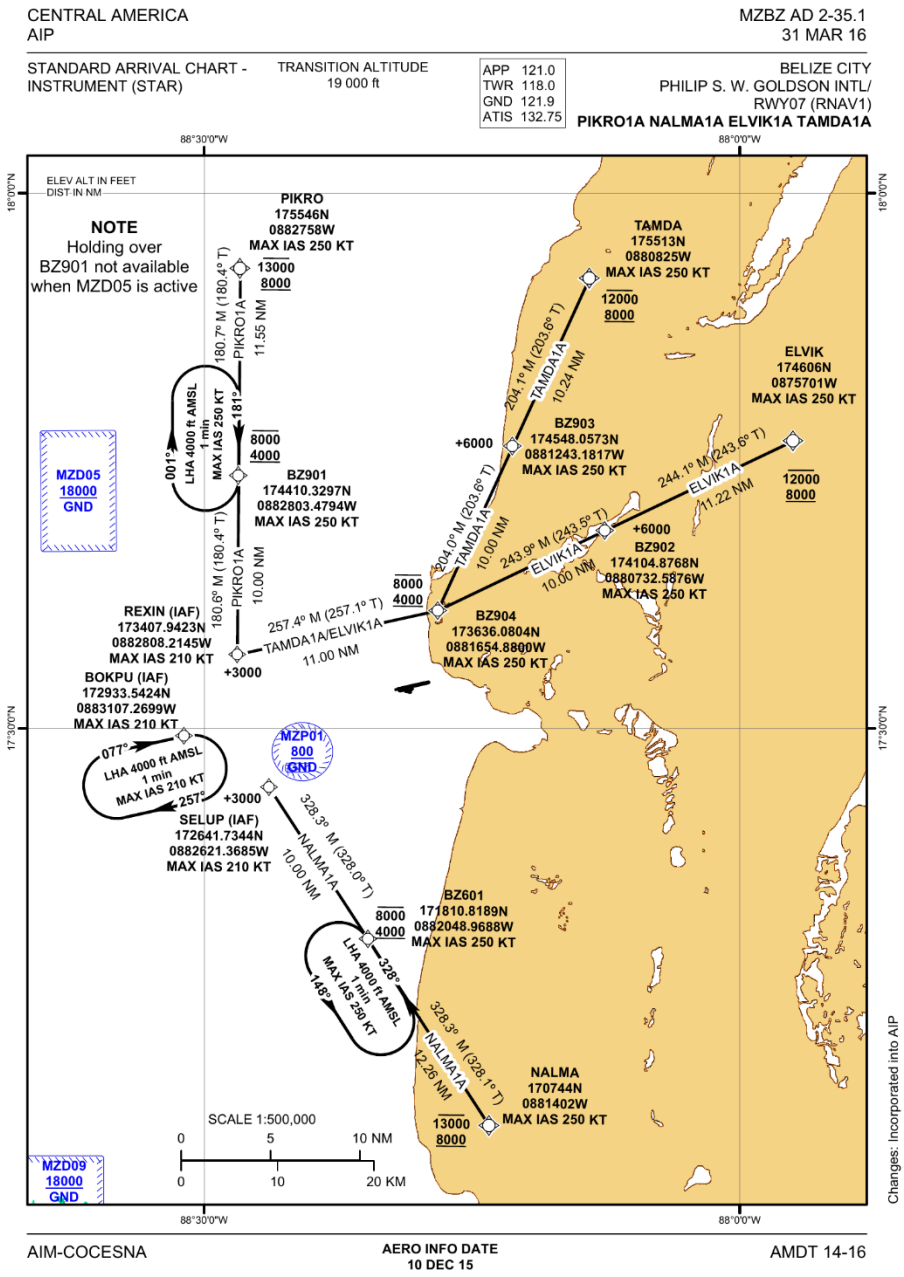

Figure 4: STAR Chart

STARS give routes to transition aircraft from en route procedures to a fix in the terminal area from which an instrument approach can be conducted and are also not always printed to scale.

These charts are used to aid in navigation and include information for various aeronautical situations that pilots may encounter. The total number of these charts is voluminous, and up-to-

date versions are published every 56 days. Charts generated from Belarus, Belize, Brazil, and Taiwan were provided. Of these charts, a sample of 902 was selected from the due to containing image data which was not duplicated elsewhere for the training sample. The training data was not weighted to be equal numbers for each chart type and instead was proportional to the distribution of provided chart types.

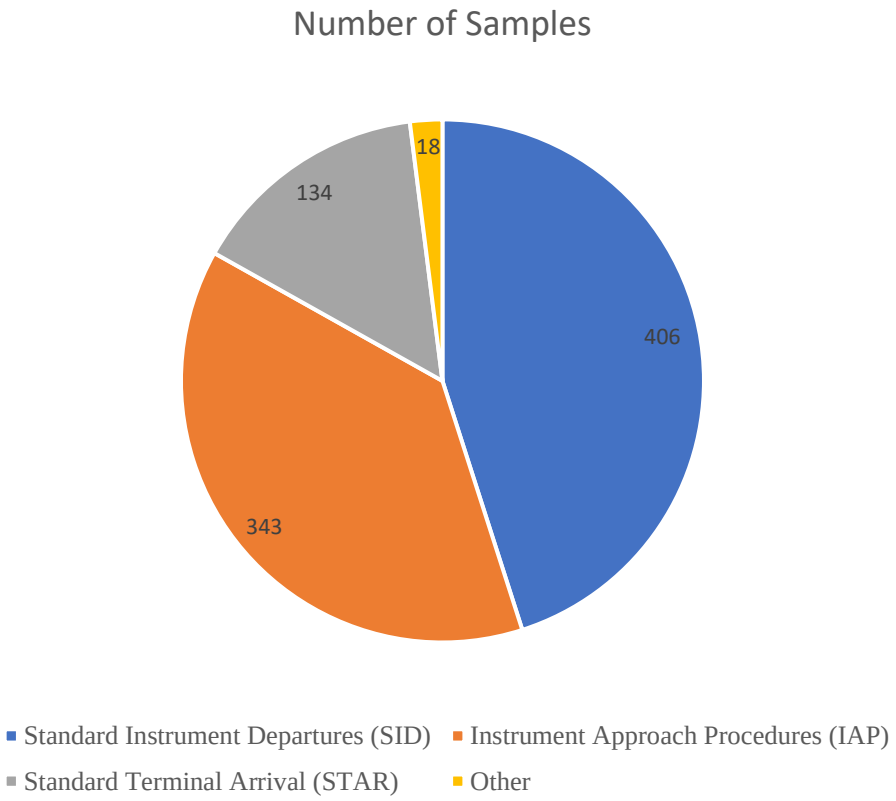

Figure 5: Illustrating the distribution of chart types in the training sample.

The largest portion of charts used was 406 SIDs making up roughly 45% of the sample. The next largest portion was 343 IAPs which made up roughly 38% of the sample. The 134 STAR charts made up the third-largest group making up roughly 14% of the sample. The remaining charts were 18 standard departure charts and an air traffic controller minimum altitude chart. Some charts may be classified as a sub-category, such as some of the Airport Diagrams were included as IAPs. Some charts, specifically aerodrome obstacle charts and aerodrome obstruction charts, were also included initially. These were later removed as they were roughly 141 of them in the initial sample, but the very high-resolution images, when resized, would lose many of the fine lines and other features, which would be necessary for analysis as the requirement for uniform input necessitated resizing to a smaller resolution than originally supplied.

3.1 Labeling Methodology

Creating a supervised learning situation for the Mask-RCNN model would require labeling the data. The choice of how to label the data is an important consideration. Here the problem would not be to detect a specific object within a chart or a series of objects, but instead, find the common sorts of features which made up these charts in general. To this end, two main labeling schemes were tried. The first was to annotate each chart's image sections as the classifications given here (IAPs, Airport Diagrams, DPs, STARs, and CVFPs). This proved challenging to classify, which likely is due to these charts' similarity when comparing one to another. An example of a Standard Instrument Departure chart has many similar features when compared to a Standard Terminal Arrival chart. After some preliminary training and testing, these annotations were removed and replaced with a much simpler two-class classification system. Images were annotated by bounding boxes that encompassed the charts' regions, which had visual chart data as image regions, and the pure text sections were classified as background regions, as shown in Figure 6.

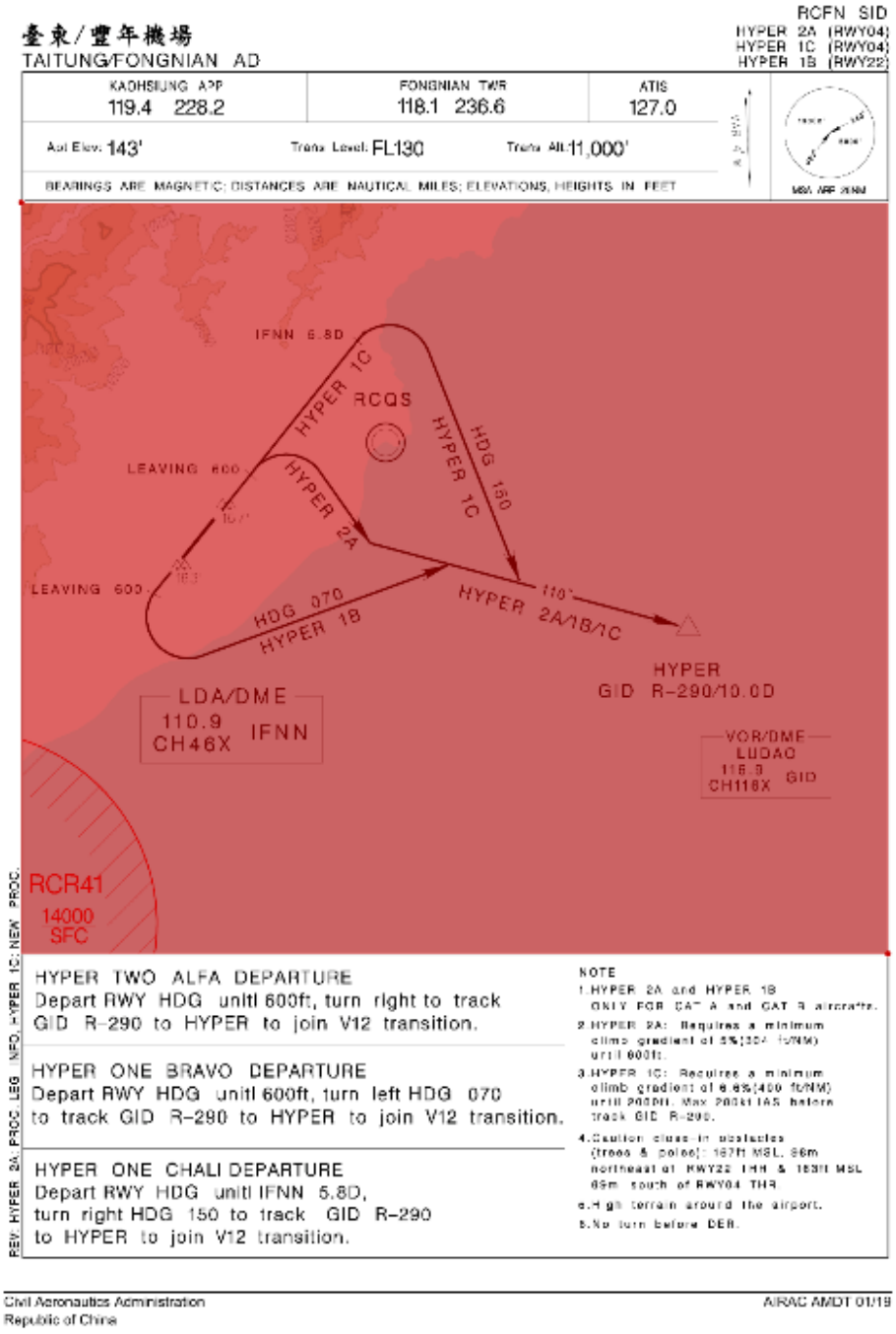

Figure 6: Annotated image with the marked region in red.

3.2 Labeling Challenges

The input charts were supplied by Portable Document Format (PDF) files. Building the training data set started with creating a small program that converted from these PDF to image files. An image file was specifically created from each page of the input PDF. Then these image files were labeled using LabelMe, which had support for creating image masks from annotations [29]. These masks would then be used as input to the artificial neural network. The image annotation data was stored in separate JSON files for each chart page. While initially, each label was for the different chart type, in the end, this was replaced with the same label for all chart types to convert this to an image or background classification system. This was done to simplify the image region detection process, as the various image regions were increasingly misclassified as background or erroneously broken into more image segments than the present. Another issue that arose was in the mask classification, as while the images were being classified under this original mixed class classification system, the masks were more spread out and, depending on the chart type, would have different polygon regions annotated. After initial training, this was revised to have a more uniform label scheme (image or background) and uniform polygon regions. These new simplified annotations were then used in all further training.

3.3 Change Detection Performance Evaluation

Evaluating results is challenging because of the difficulty in creating some ground truth datasets [3]. Being precise about what has changed, if that apparent change is not the resulting noise, and if the change is significant are all parts of defining a system to detect differences and changes. Quantitative analysis is necessary for discovering if a given technique is effective, and here Jaccard Similarity score was chosen to evaluate the performance. The map of the predicted changed regions and the hand created ground truth dataset were compared to show where the two agreed on a change (true positive), where they agreed on no change (true negative), where the prediction had a change not present in the ground truth (false positive), and where the prediction did not have a change present in the ground truth (false negative). The model should be biased towards minimizing false negatives while correctly detecting as many true negatives as possible. Incorrect identification of false positives is slightly less of a concern as a further manual review can eliminate these incorrectly identified changes.

When treating each pixel in a chart as a classification problem, such that each pixel is either a true positive, false positive, true negative or false negative each chart's proposed changes can be examined in more detail. Something like a true positive rate (also known as sensitivity) can be calculated to show the probability that an actually changed pixel will be identified as changed, the true negative rate (also known as the specificity) will show the probability that an unchanged region will be accurately identified as unchanged, precision can be used to measure what fraction of the detections are changed, and the false-negative rate can show the chance that a true positive would not be detected [23].

Table 1: Performance Evaluation Metrics

Name of Evaluation Metric:	Definition:
True Positive Rate (Sensitivity)	$\frac{\text{True Positives}}{\text{True Postives} + \text{False Negatives}}$
True Negative Rate (Specificity)	$\frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$
Precision	$\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$
False Negative Rate (FNR)	$\frac{\text{False Negatives}}{\text{False Negatives} + \text{True Positives}}$

Chapter 4 – Methodology

4.1 Overview

The change detection model proposed here begins with one original chart which will be compared to a potentially modified chart. The charts are both resized to a uniform resolution to be compatible with the Mask R-CNN neural network. Instead of predicting image or non-image regions, this network is used to extract low-level features. These features' presence or absence are then compared for each region of the image. A substantial difference is detected among multiple features in each region, then marked as potentially changed. These potentially changed regions are then marked on both the original and modified chart to show which regions require examination.

4.2 Feature Extractor Training Input

The annotated charts would be condensed into a single compressed file, including the image of the chart's full-page, a set of masks for each annotated image region, a set of labels for the image region, and the image's original dimensions. These images and annotations were then augmented to produce additional training data. For each hand-annotated image, an additional eleven augmented images were generated using different combinations of flipping horizontally, flipping vertically, adding gaussian blur, and adding padding to different regions of the image. This was done using the imaging library, which had extensive support for augmenting image masks [30]. This would allow an expanded set of images to train on for the artificial neural network to find more general features instead of just focusing on the specific regions in the annotated images. The after-augmentation total, the total of training images with annotations, was 10,824. Getting the best features without over-fitting for the training set would require a thorough analysis of when the feature extraction would best detect the general features. Ten-fold cross-validation was used to decide when to stop training.

4.3 Using Convolutional Neural Network Layers As A Feature Extractor

The process begins with the framework of the Matterport's implementation of the Mask-R-CNN model using Keras and TensorFlow [31] [32] [33]. The idea here is to use the Mask-R-CNN network as the feature extractor for image features across multiple convolutional layers and use these features to compare two similar images. Instead of relying on the final output of these complex networks, the goal was to annotate a variety of available charts to teach the network to find those features that indicate the presence or absence of image features and use the relevant features to compare. To this goal, training the network's weights to discover features was done using augmented images and ten-fold cross-validation to discover with Epoch to stop on. The data would begin with weights produced by training on the COCO benchmark dataset using transfer learning so that the features would already be roughly detectable from the start [34]. These already supplied weights were generated using the ResNet 50 classifier, which would be used as the basis for training this model. To create uniform input the charts were resized to be 768 by 768 pixels. Empty space was added to whichever (height or width) was smaller to preserve the original image's aspect ratio.

The Resnet 50 classifier is an architecture for deep neural networks made up of 50 layers split into stages. The first stage takes the input of the image and pads it with zeros (3x3 padding), then convolutes it with 64 filters of size 7x7, then applies batch normalization to the results. This produces an initial feature map. Then the rectified linear activation function is applied to the values in the feature map as an activation function to introduce non-linearities. Then a 3x3 max pooling layer with stride 2x2 is applied to pool the results into smaller regions. After this first stage, the following stages are a mix of identity blocks and convolution blocks. The two different types of blocks are similar except for the fact that the convolution block has a convolution layer on the shortcut, and this shortcut is the foundation behind the ResNet classifier [35].

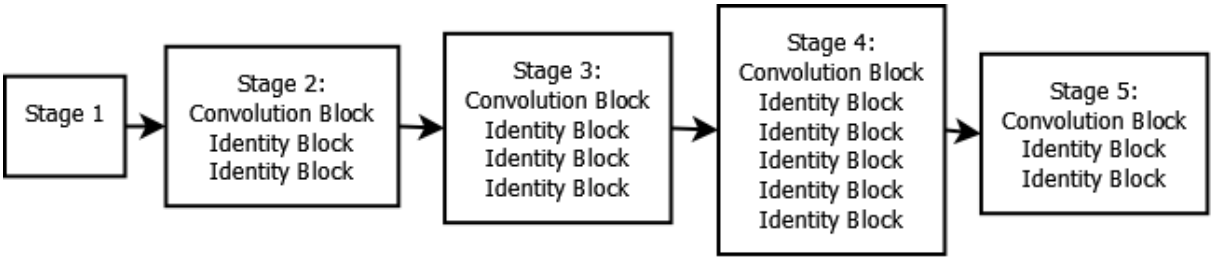

Figure 7: The Implementation of all the blocks in Resnet 50

The idea here being that while deep models are useful as adding layers better extracts features. This introduces a problem where training is more difficult since the expected result is very disconnected from hidden layers. Normalization can be used to account for the vanishing and exploding gradient problem (which caused those issues with converging to a solution due to those hidden layers). However, just adding more and more layers did not always lead to better solutions, as when these deep networks start to converge, it appears as though a shallower network may have better results [35]. The solution proposed would allow a behavior similar to the shallow networks while still including the potential benefits from more layers. This is done with what is referred to as a residual block. The idea is that there is some underlying mapping H(x) (where x is the input) and a series of stacked non-linear layers fit the mapping of the function of the residual. Rearranged, the underlying mapping H(x) can be discovered as being equal to some function F(x) + x. The model asserts that it is easier to optimize this underlying mapping rather than having to optimize the original mapping. This formulation is modeled in a feedforward neural network with those shortcut connections where one or more layers are skipped [35]. This is done to perform identity mapping where the residual outputs are added to outputs of the stacked layers.

Practically this allows the propagation of gradients to initial layers in ways that these layers can learn quickly in a similar fashion to those final layers, which are more closely tied to the error. When differentiating the identity block and the convolution block, the idea is that some blocks are made up of a one-by-one convolution layer, then a three by three convolution layer, and then another one-by-one convolution layer. The first one by one layer reduces the dimensions of the image, then the three by three-layer has smaller input and output dimensions, then the last one by one layer restores the final output from the block to be the same dimensions as the input. This is done to decrease the computational expense of the three-by-three convolutions. So, the stage one input is then fed to stage two, which is made up of a convolution block, then two identity blocks. The output of stage two is then fed to stage three, which is made up of one convolution block and three identity blocks. Then stage four is made up of one convolution block and five identity blocks. This is then fed to stage five, which has a convolution block then two identity blocks.

Finally, the results of stages two through five are used as inputs for the Feature Pyramid Network (FPN). The concept behind an FPN is that it is designed to use the pyramid-like feature hierarchy of a convolutional neural network. It takes a single-scale image of arbitrary size as input and outputs proportionally sized feature maps in a convolutional fashion [36]. The FPN uses a feedforward computation of the backbone of the convolutional neural network, here ResNet, which results from the last of each of the previously noted five stages in residual blocks (while discarding the first stage's results). The FPN also generates higher resolution features by upsampling spatially coarser but more meaningful feature maps from higher pyramid levels. The results of these up-sampled feature maps are enhanced by the feedforward generated features using lateral connections. These take the feature maps of the same spatial size from both the normal feedforward features and the higher resolution up-sampled features and merges them together. Then the outputs of this are combined together so that this new feature set have a consistent number of feature dimensions and do not include non-linearities.

The final portion of the model is computed using the final RoIAlign layer, which calculates regions of interest and aligns the features detected with the input. This is based on the previous RoIPool method, which takes a small feature map from each region of interest. The point here is to keep the features detected to remain corresponding to the pixel-accurate masks. After initially getting a set of weights from the COCO dataset, the entirety of the model was set to be trainable during the training phase. Learning momentum was set at a level of 0.9, and the learning rate was

set to 0.001, similar to previous performance in other examples with the same Mask R-CNN model [24] [31].

4.4 Mask R-CNN Output

After training the complete model on the training samples and testing to find the stopping point using ten-fold cross-validation, the 10th Epoch of training was used. Since the training data was already pre-trained on a much larger dataset, the previous training would teach some of the basics of the sort of features which we are searching for. The trained neural network would then be used to create a variety of feature maps to indicate the presence or absence of a feature in the region specified by a pixel and its nearby neighbors. These features would be those features, which best indicate that an image portion of a chart is present at that pixel and allow for a higher-level representation of that image region more so than merely a pixel to pixel comparison would allow. The activations of these feature maps would then be useful to compare the presence or absence of a particular feature in two separate images. This would allow a rough approximation which has some of the advantages of a pixel to pixel comparison while using a feature to feature comparison.

While training the neural network to detect features was done in a supervised fashion, the network's final results would then be used in an unsupervised fashion. The amount of properly labeled ground truth comparisons is much lower than the number of images, which could be labeled. The idea here is to leverage a large amount of trainable data to improve the feature extractor, then using more conventional algorithms to generate results to compare. To that end, once the training was complete, the workflow for creating comparisons was as follows.

Two PDFs of comparable charts are supplied. The first will be referred to as the base image, the second the modified image. The charts are read in as image files. The two charts are supplied to the previously trained neural network one after the other. This produces two separate outputs of all the feature maps at all Mask R-CNN stages (the ResNet backbone, the Feature Pyramid, and the masks). The various feature levels were compared to find which level of the model had sufficient resolution in terms of the features present and enough abstraction. In the end, the output of the third stage was chosen.

The output of the third stage has a variety of filters. The output of the base image's feature map at the end of that stage is then compared to the modified image by subtraction. The base image's absolute value minus the modified image is then used as the feature map of the difference.

To generate a composite of all the layers, their activations are then averaged. This produces an array, which roughly indicates where, on average, the features of a given pixel substantially differed. This array is then upscaled to match the same resolution as the base and modified image.

The array values which are greater than zero are then used to indicate which pixel regions have a high likelihood of being changed. This technique allows for a certain amount of shift shift-invariance, which is beneficial and further emphasizes those pixels whose features have been learned as the most relevant for differentiating between image and non-image regions. For each example evaluated, a ground truth mask is manually generated by creating a black and white image with the unchanged regions marked in black and the changed regions marked in white.

4.5 How To Evaluate Results

Three techniques are evaluated here. The baseline technique will be a simple pixel to pixel comparison, where both the original and modified image of each pixel's values RGB values are converted to a single grayscale value. The absolute value of the modified image's grayscale value is then subtracted from the original image to create a new black and white image where both images dissimilar regions appear on this new image. This is used as a pixel to pixel comparison. For each set of charts, this baseline will be evaluated against the manually generated ground truth. This generates a black and white image which is then compared to the ground truth image. This is evaluated using the Jaccard Similarity score. This takes the total of the pixels where the two images are both marked as changed and divides it by the total number of pixels where either image is marked as changed. The closer the Jaccard Similarity score is to 1, the more accurate the prediction is. The structural similarity score generated change predictions are also similarly generated to create an image where that algorithm predicts changes occurred. Likewise, the neural network's feature difference generated image is changed into a black and white image where black regions are non-changed and white regions are changed.

Using the evaluation metrics described in Chapter 3, the proposed changes can be evaluated. Each set of proposed change regions is treated as a series of changed or unchanged pixel regions. Aggregating across a small sample of 16 hand-annotated change charts allows all the pixels in all the images to either be correctly or incorrectly classified. The metrics will be used on a per two chart change comparison basis but also aggregated across all the examples.

Chapter 5 – Results and Discussions

It is especially worth noting that part of this is to detect significant image changes. This task aims to show where a change in the meaning of the chart has occurred. The idea of significant changes includes within it, in this context, the idea that the changes should be in the content of the image as it may be used. A simple shift in the margins of the chart that does not change the contents of the chart entirely, just the location where the chart shows up on the page, should be accounted for. The changes which are focused on should, for example, be something like a change in the location of a runway, a change in the angle of a turn, or a move from one location of approach to another location. Ideally, just these changes would be detected. This is challenging, and the threshold for differences detected can vary.

5.1 Examples of Area Charts with small changes and shifts

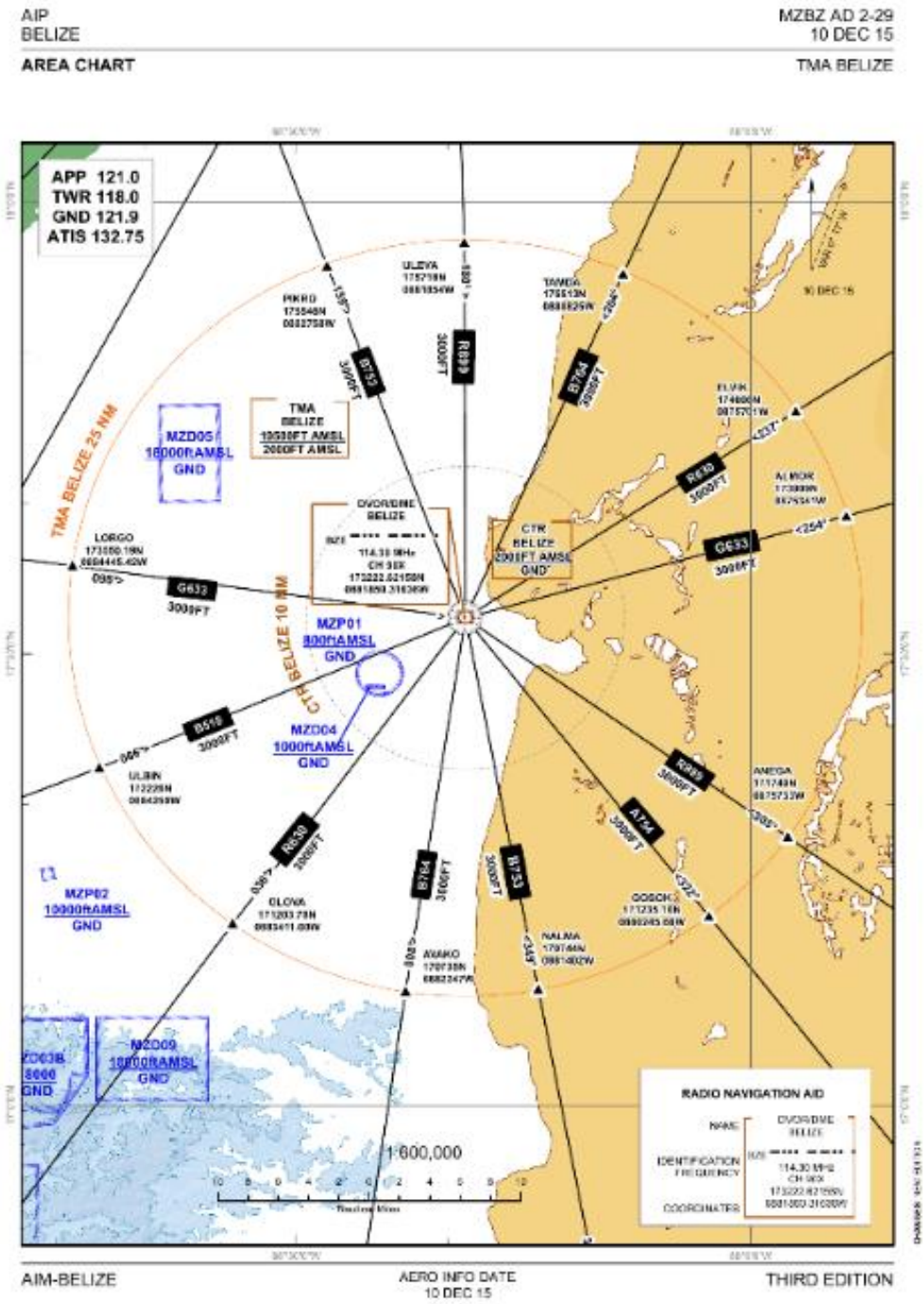

Figure 8: Instrument Approach Procedure Chart (Original)

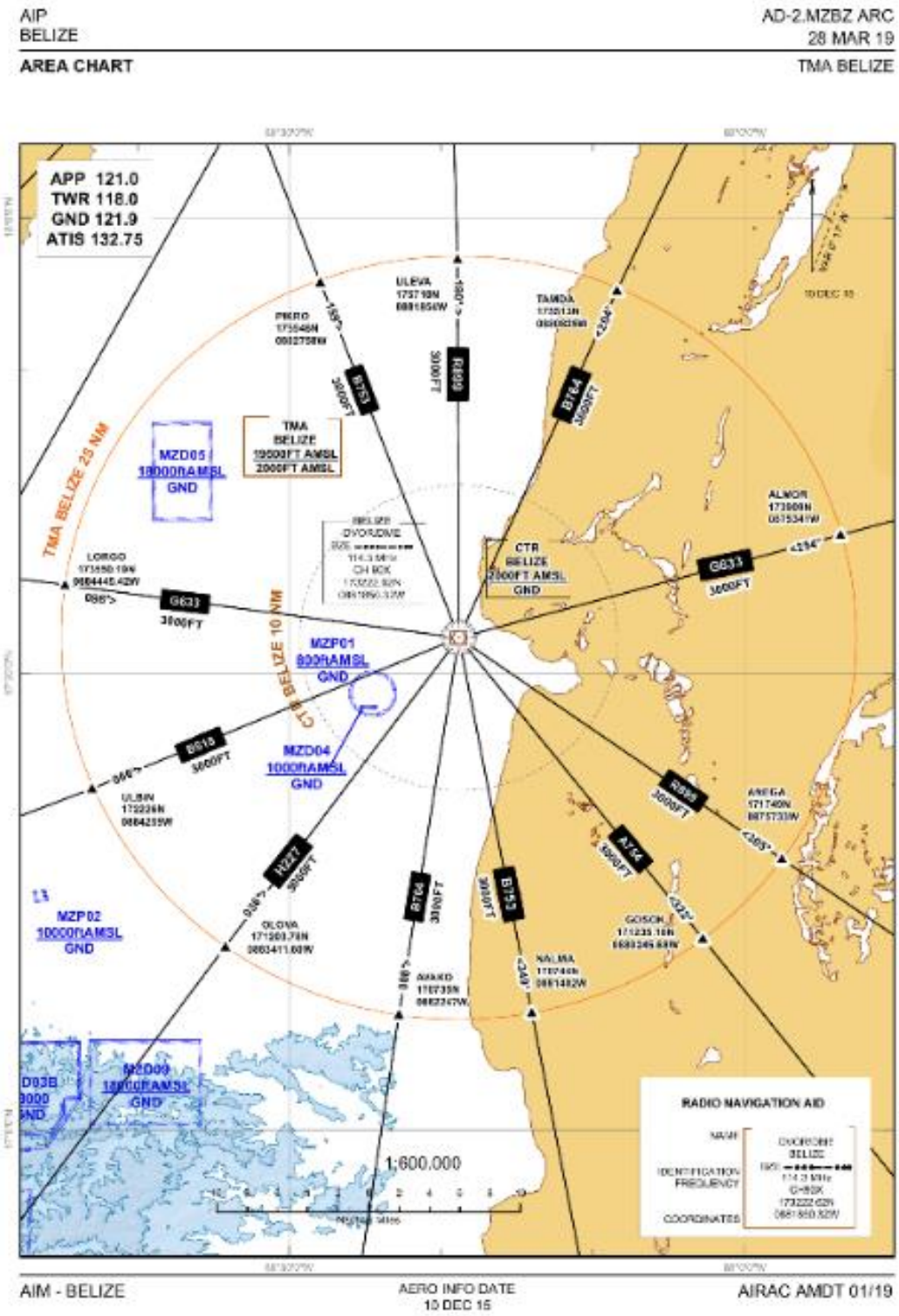

Figure 9: Instrument Approach Procedure Chart (Modified)

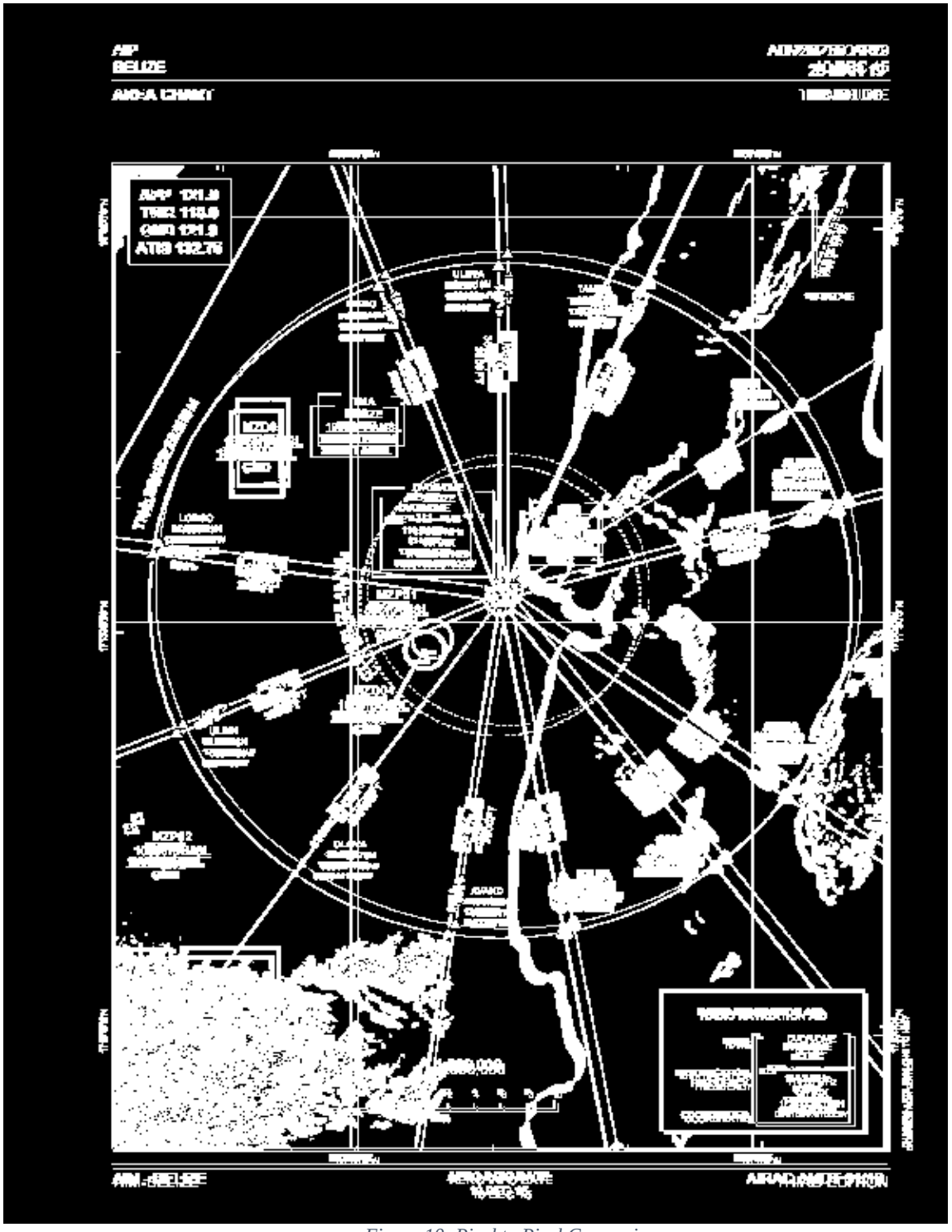

Figure 10: Pixel to Pixel Comparison

While charts' contents may remain similar, the image may shift slightly in different directions. Any direct pixel to pixel comparison will detect the shifts as changed regions. This will overestimate the amount of change.

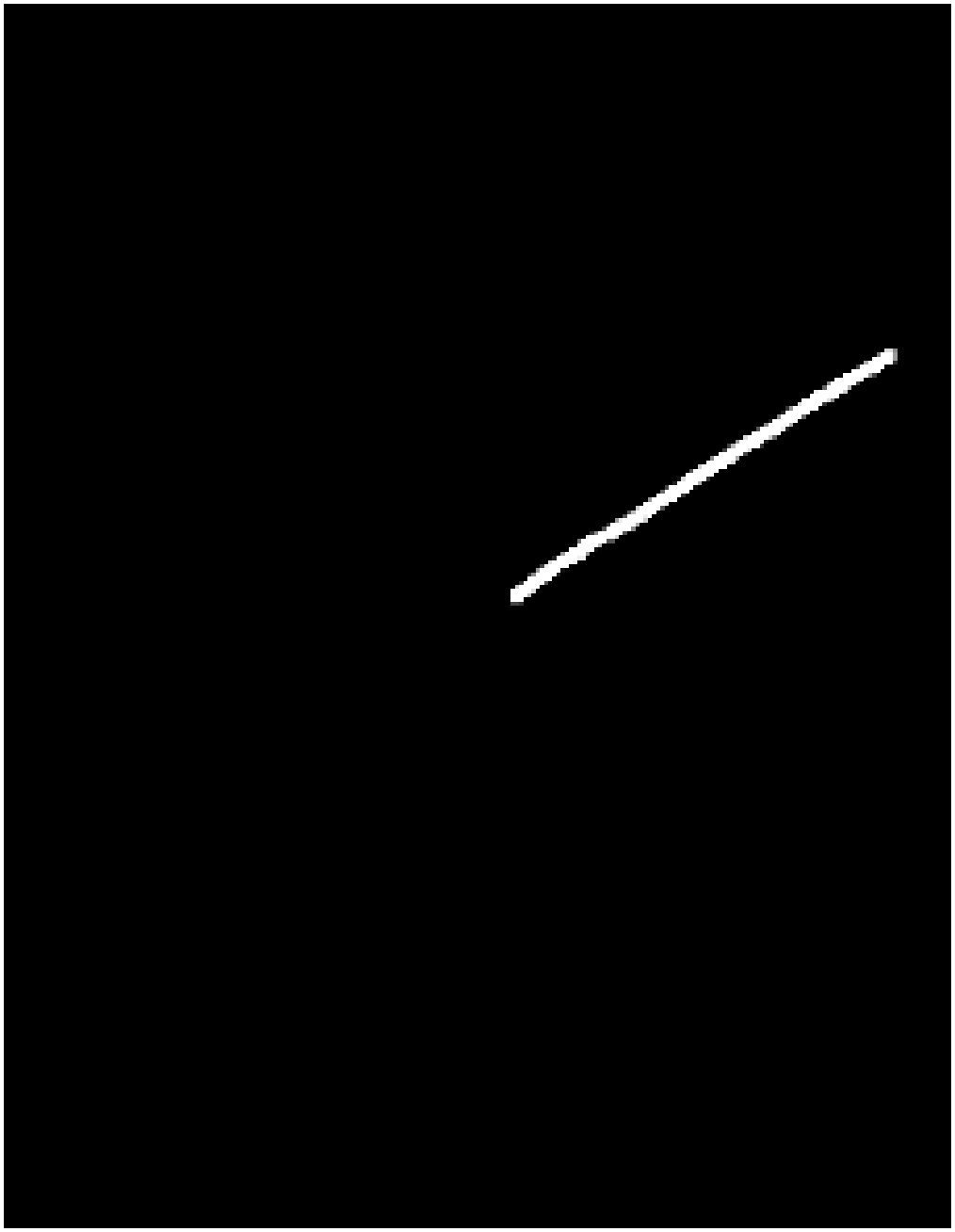

Figure 11: Hand Annotated Ground Truth

Instead of overestimating the change, however, the system should ideally detect the main addition of a line in the upper right-hand corner. None of the shifted regions have been changed substantially. So instead, just a small line should be detected.

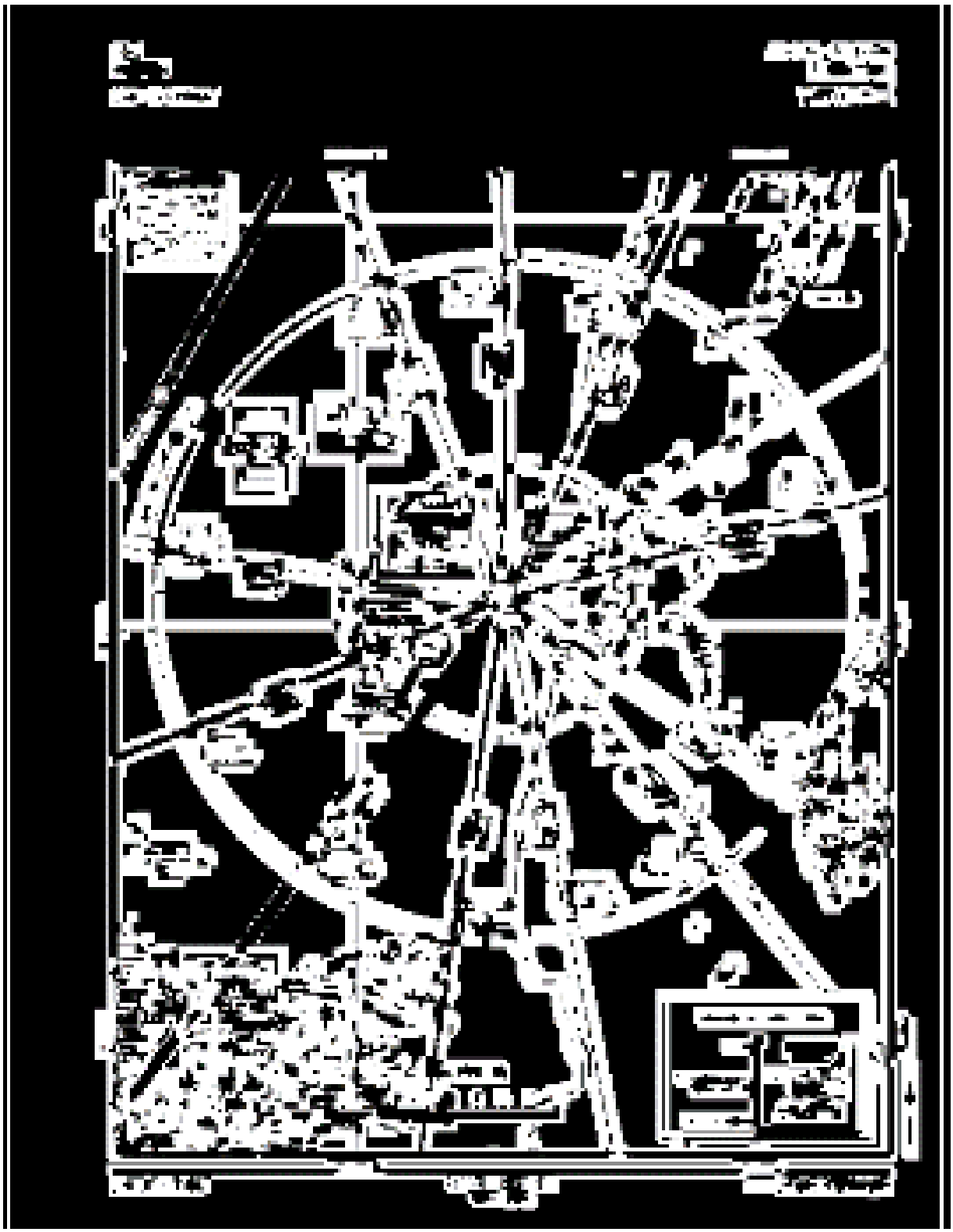

Figure 12: Structural Similarity Index based changes

Like the baseline comparison, using structural similarity to show the regions with changes based on the different thresholds would also indicate large numbers of changes.

The proposed changed pixels from the output derived from the neural network's feature extractor also detects some of the shifts as changes, but less so.

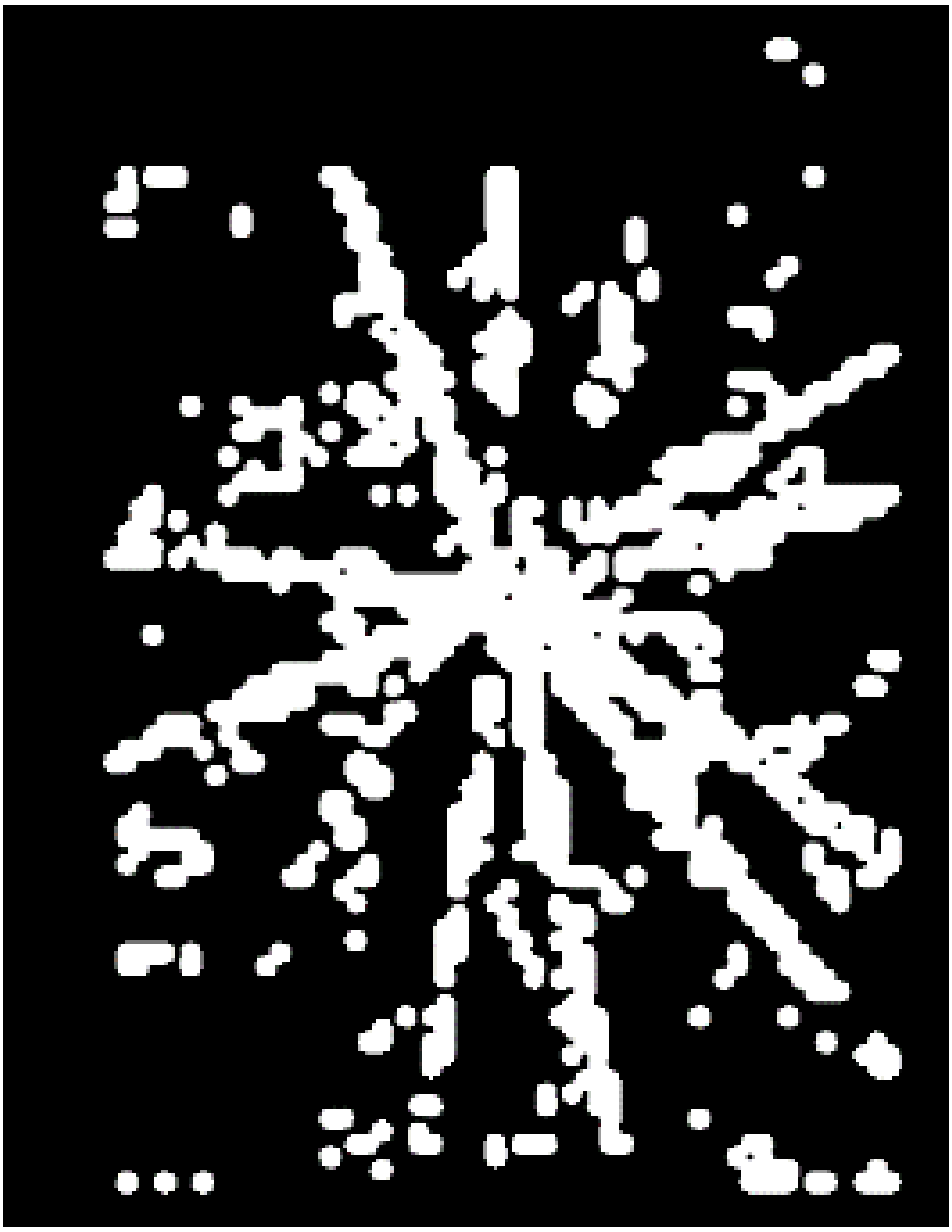

Figure 13: Neural Network Feature-Based Changes

Each of the charts can detect the addition of the line as marked in the ground truth. Their Jaccard Similarity scores are all low, however, as there are large amounts of proposed changes which are, in fact, falsely detected. Superimposing the predicted change regions can help illustrate what the regions of these changes look like.

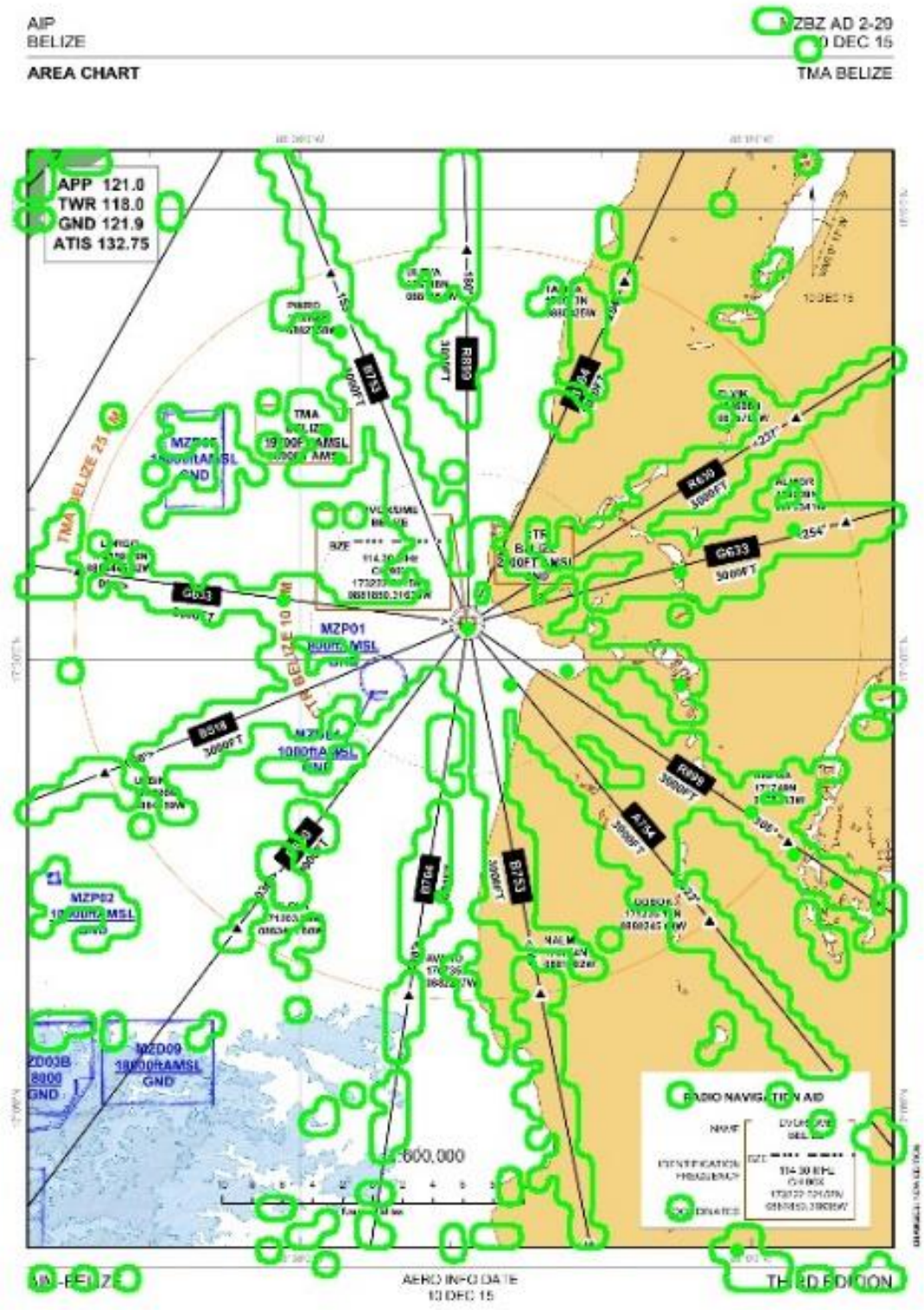

Figure 14: CNN Changed Regions Circled in Green

As shown in Figure 14, we can see that the artificial neural network has proposed a system, which is closer to the simple change proposed. The region in the lower left-hand corner, for

example, is not shown as having changed as much in the neural network example as in the baseline or structural similarity. A similar pattern emerges when investigating other charts. At times the number of false positives is high but lower than in the other proposed methods. When the shifts were minor, for example, the system performed well at not indicating a change when only a shift was present when compared to the baseline approaches, which can be seen in Table 2.

Table 2: Comparing Results for Techniques on a pair of example Instrument Approach Procedure charts

Model	Sensitivity	Specificity	Precision	FNR	Jaccard
CNN	0.943064578	0.785665858	0.024799035	0.056935422	0.024761962094532904
SSIM	0.862820023	0.659057833	0.014415312	0.137179977	0.014382348819730188
Pixel to Pixel	0.57585021	0.770984106	0.014324142	0.42414979	0.014174591081388677

The sensitivity score here shows the ratio of correctly identified change pixels and all the manually indicated change pixels in this particular image. Here the high sensitivity shows that most of the changed regions were correctly detected. The sensitivity indicates that most of the line which is present in the original but not in the modified version was identified as changed. The specificity also indicates that the majority of unchanged regions were correctly identified as such. The precision score shows the ratio of correctly identified changed pixels vs. all pixels identified as changed. From this, we can see that a large number of pixels are incorrectly being identified as changed, where the true number of changed pixels is much smaller. The low false-negative rate is also a good indicator that most changes will be detected, as it indicates the likelihood that a real change will be missed. This metric is one, which should be ideally low, and that is the case here as the neural network-based model generated the lowest of all the scores. The Jaccard similarity scores are low, and this is mostly due to a large number of falsely identified changes due to the shift from one chart to the next and the small number of actual changes in comparison.

Here another chart used as an example where the changes manually marked are the addition of a series of markers indicating the addition of a variety of markers from the original image to the modified one. Again, there has been a shift from one chart to another chart, but only with the addition of these small markers. Note the duplicated nearby features in Figure 12 and note how these are not being indicated in the neural network generated indications in Figure 13.

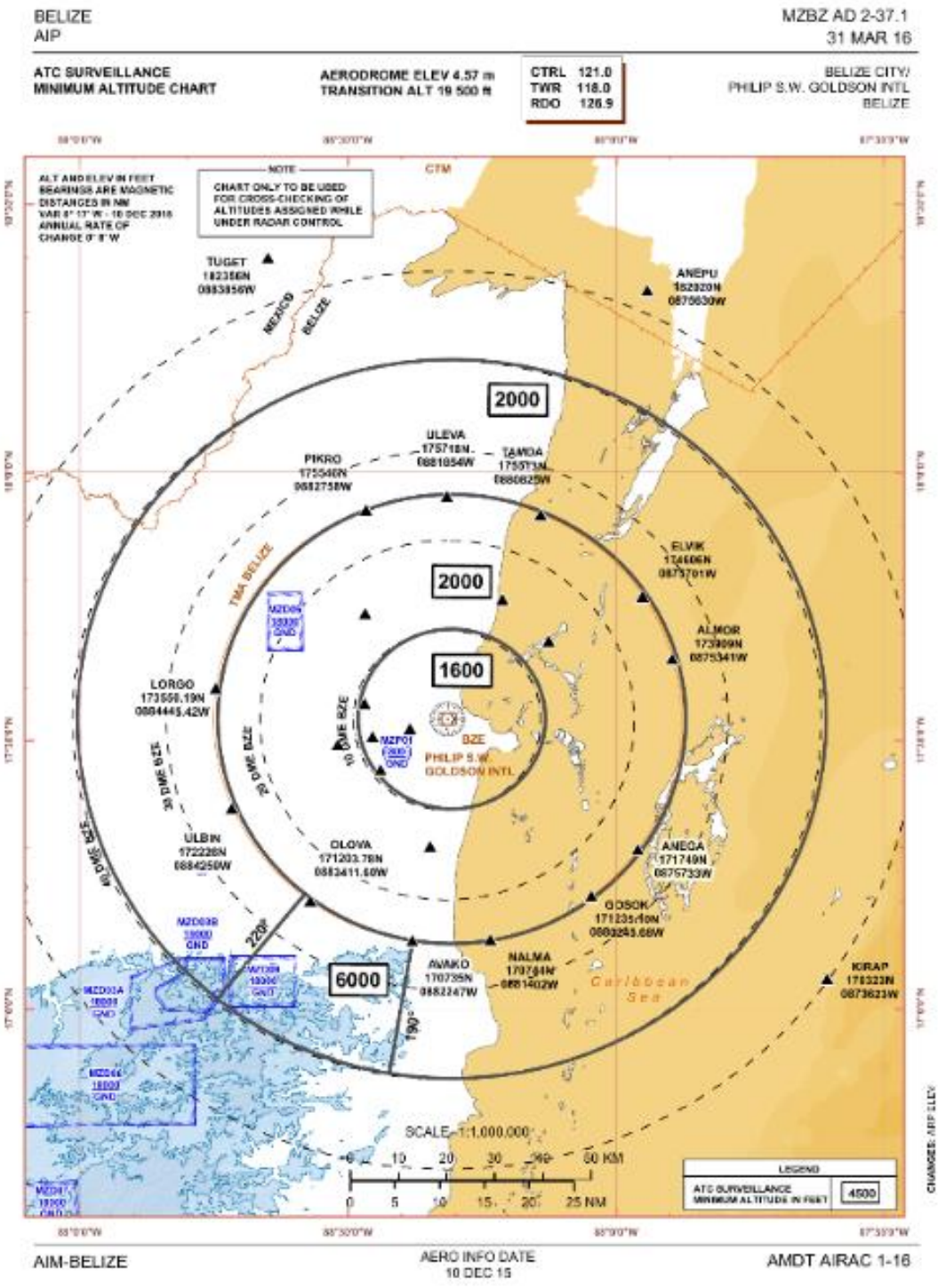

Figure 15: Example of a Minimum Altitude Chart (Original)

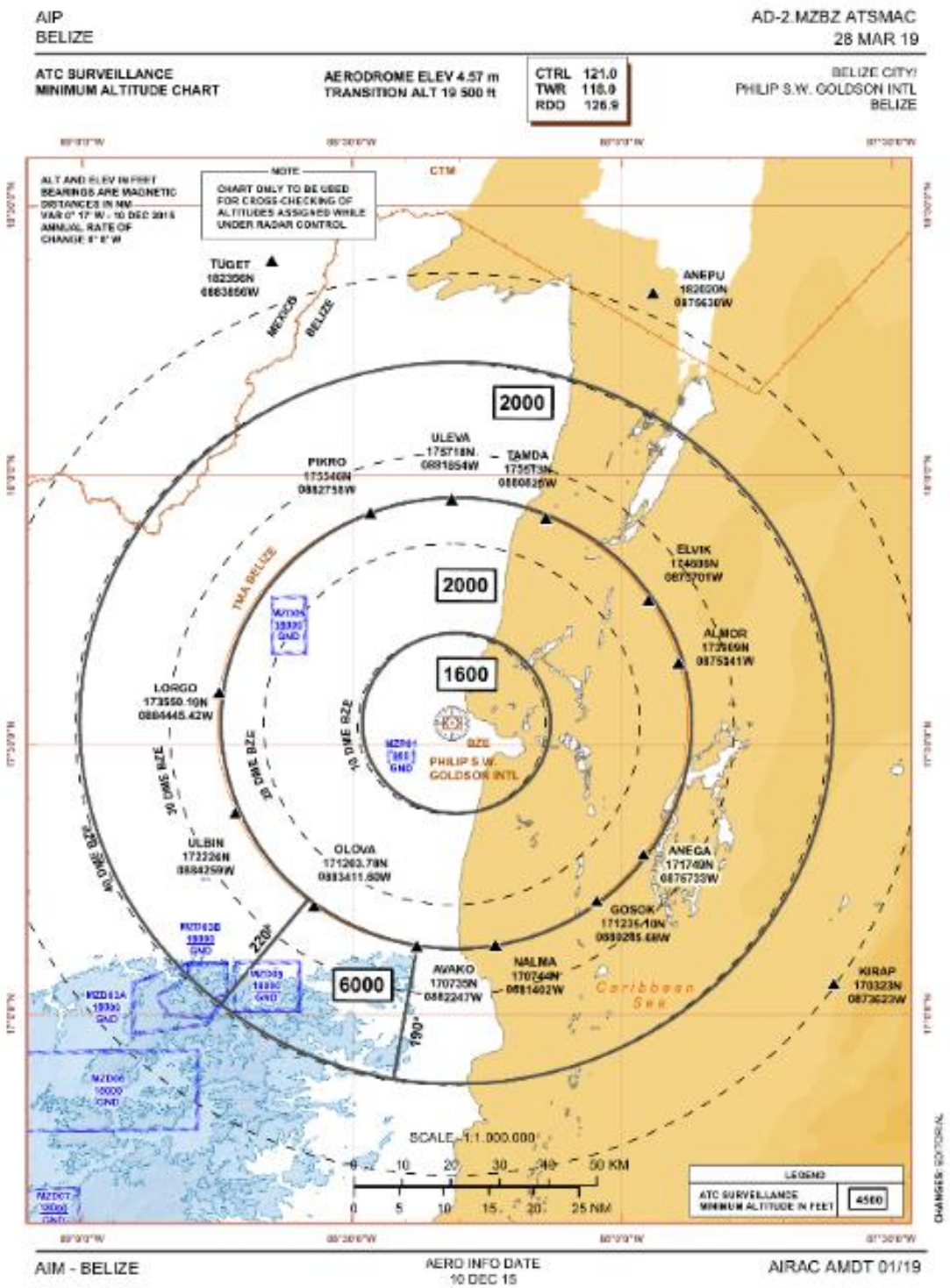

Figure 16: Example of a Minimum Altitude Chart (Modified)

Figure 17: Pixel to Pixel Comparison of the Minimum Altitude Chart

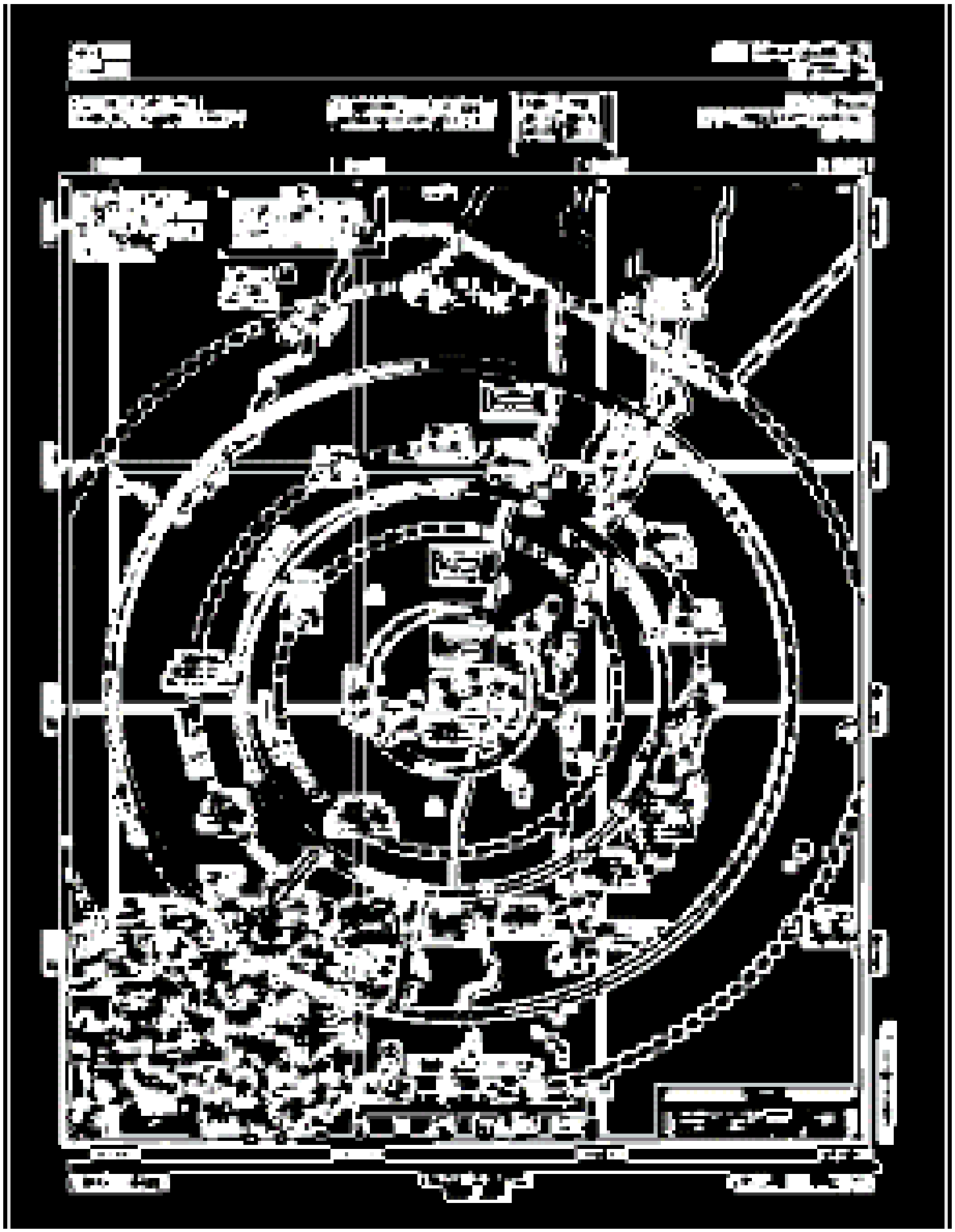

Figure 18: The results of the Structural Similarity Index detecting changed regions on the (Minimum Altitude Chart)

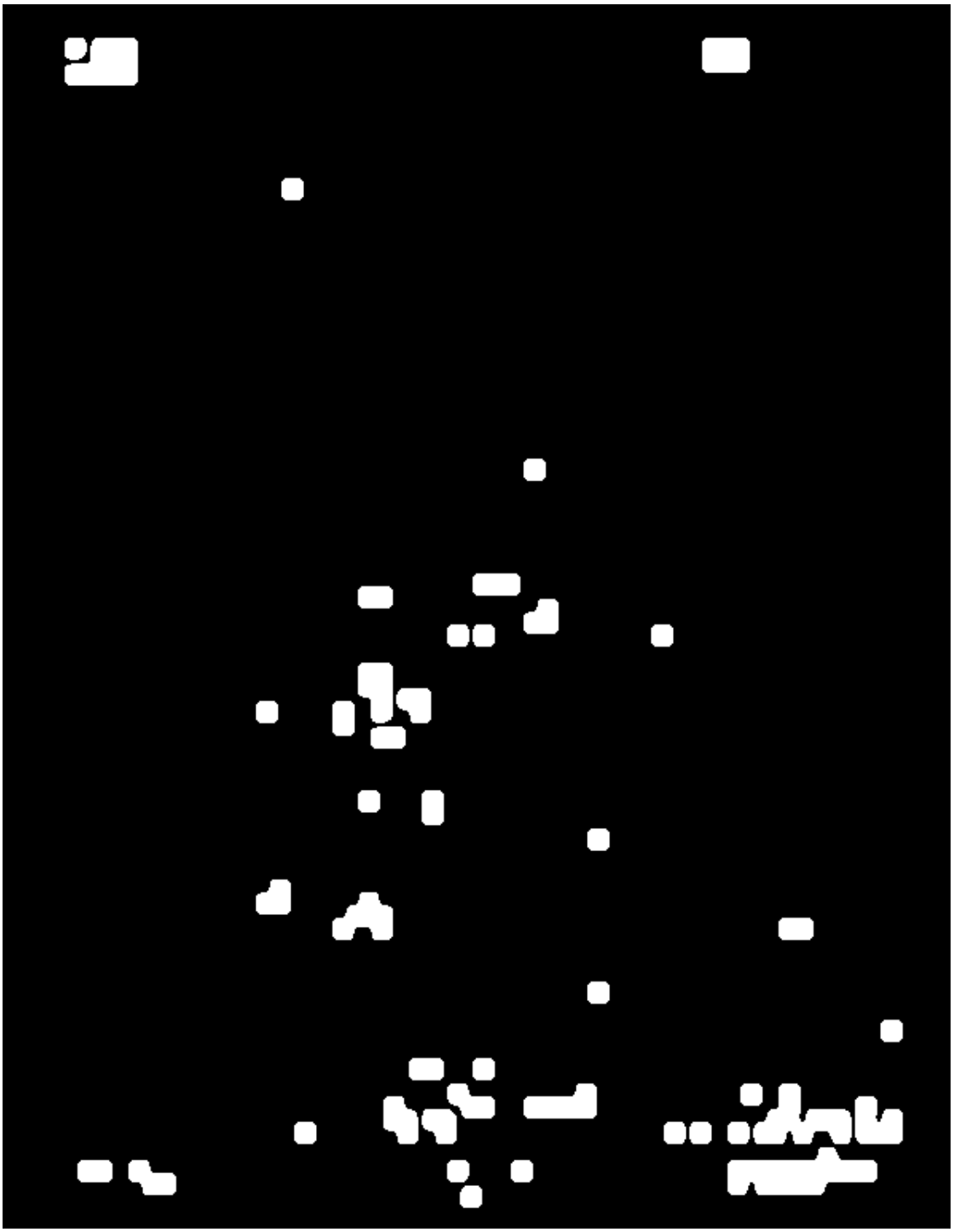

Figure 19: Neural Network Generated Changed region on the Minimum Altitude Chart

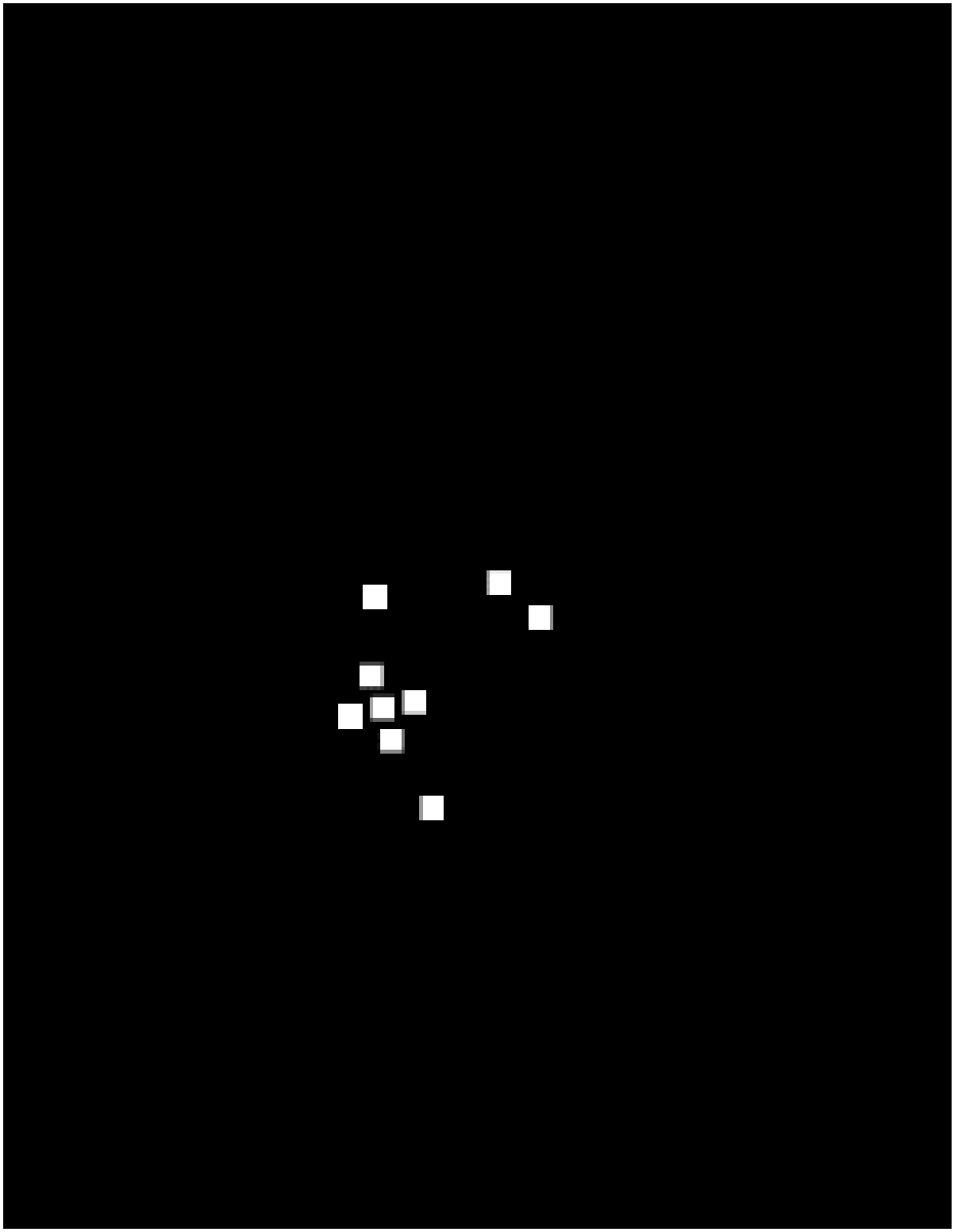

Figure 20: Ground Truth for the Minimum Altitude Chart

Table 3: Minimum Altitude Chart Model Evaluations

Model	Sensitivity	Specificity	Precision	FNR	Jaccard
CNN	0.911604938	0.96679086	0.10888286	0.088395062	0.10774528687328547
SSIM	0.648395062	0.727943552	0.010497198	0.351604938	0.0104377827065099
Pixel to Pixel	0.431111111	0.772591689	0.00836776	0.568888889	0.014174591081388677

As seen in Table 3 (and as with the previous example), we have both a high number of correctly identified pixel regions due to the high sensitivity scores, and the high specificity indicates that most unchanged pixels were also identified as unchanged. The low precision, however, does illustrate that many pixels are falsely identified as changed when they are actually unchanged. The low false-negative rate indicates that most of the pixels identified as unchanged were correct and were truly unchanged. The Jaccard similarity scores are low as the number of false changes identified is high compared to the number of true changes.

5.2 Example of Difference with Substantial Changes with a shift

When the changes are substantial, and a shift is present, the baseline model will often indicate the borders have changed. These borders, given the scale of the shift, are detected as changes in all three models.

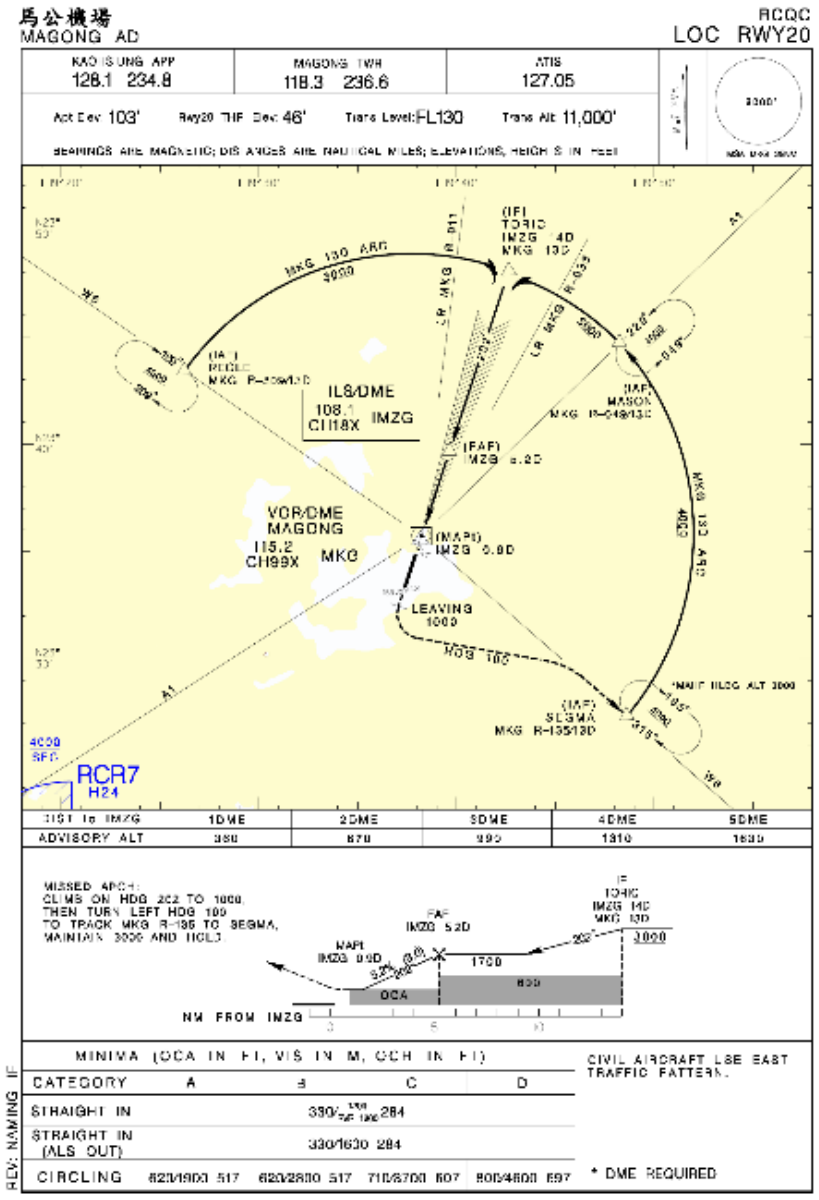

Figure 21: An example of an old AIP chart (original)

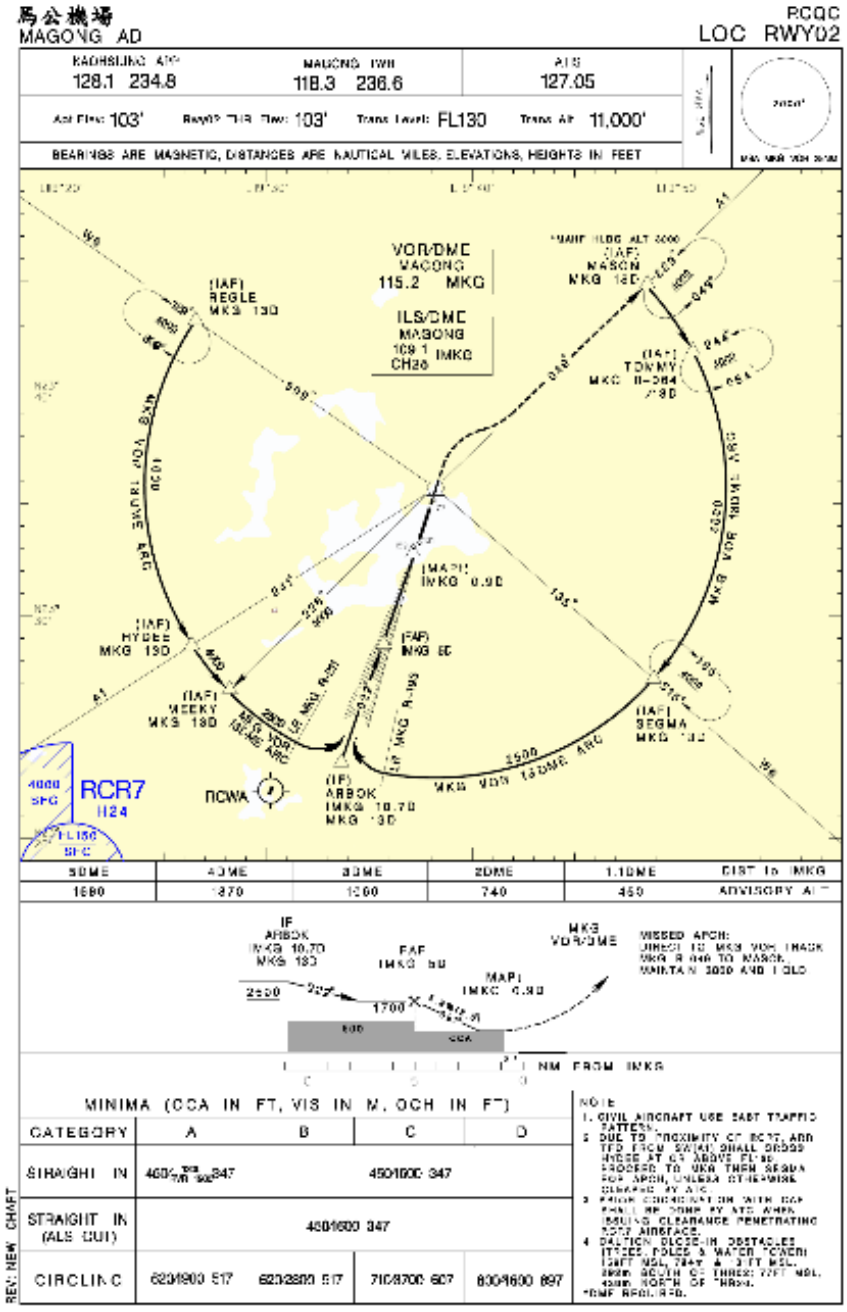

Figure 22: An example of the updated/modified version of the same chart as Figure 21

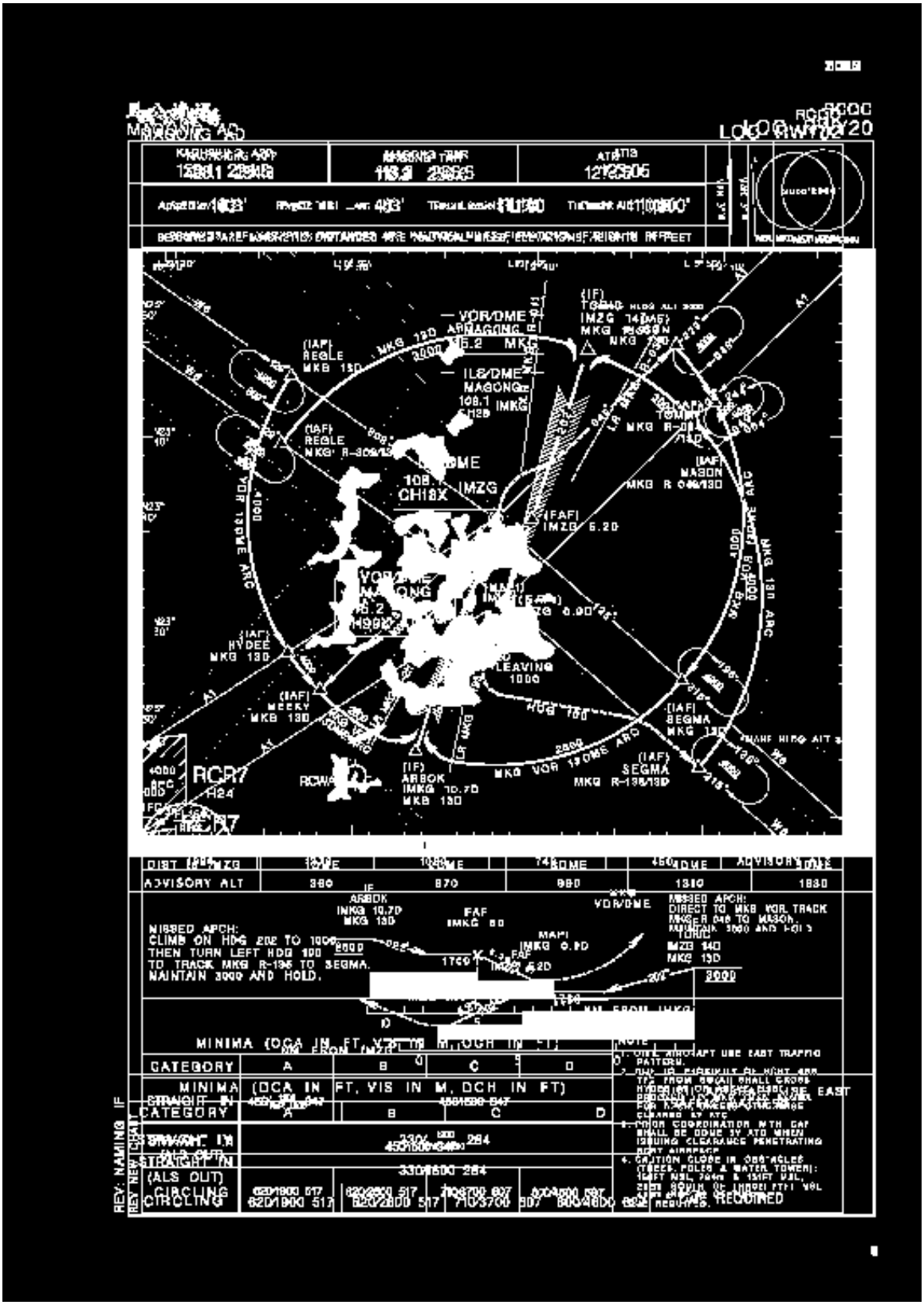

Figure 23: The pixel to pixel comparison of Figure 21 and 22

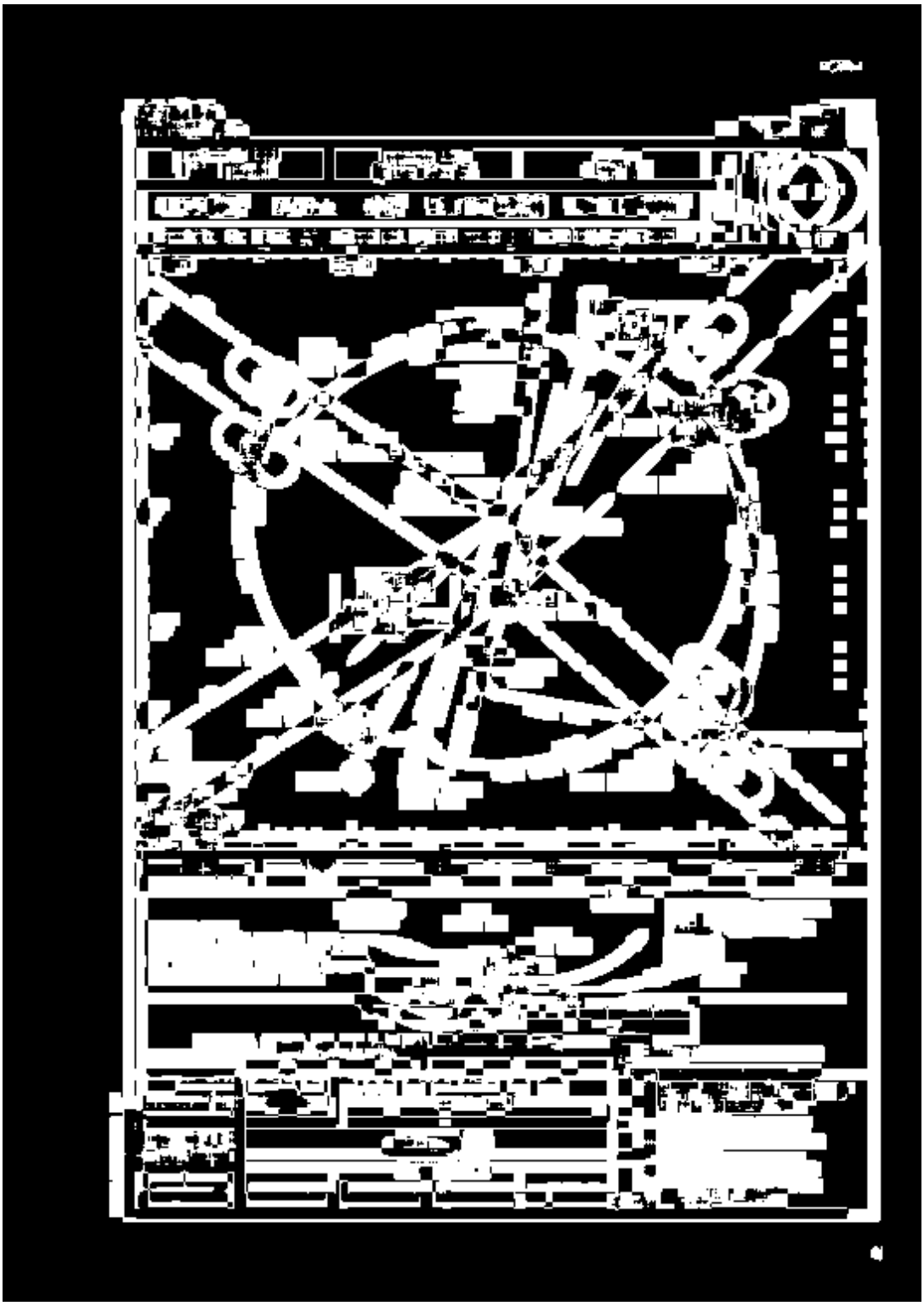

Figure 24: The Structural Similarity Index indicated differences between Figure 21 and 22

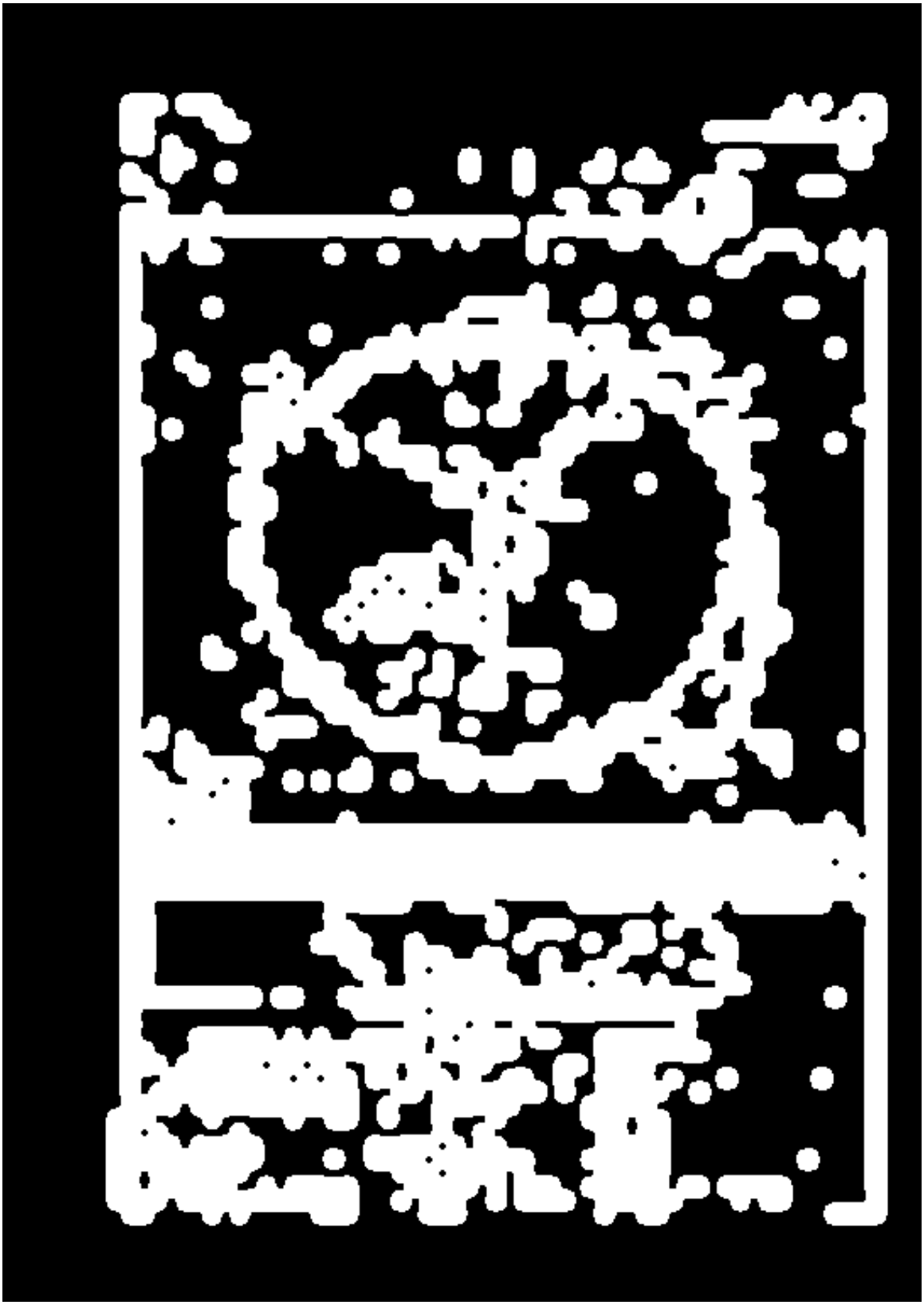

Figure 25: The Neural Network predicted differences between Figure 21 and 22

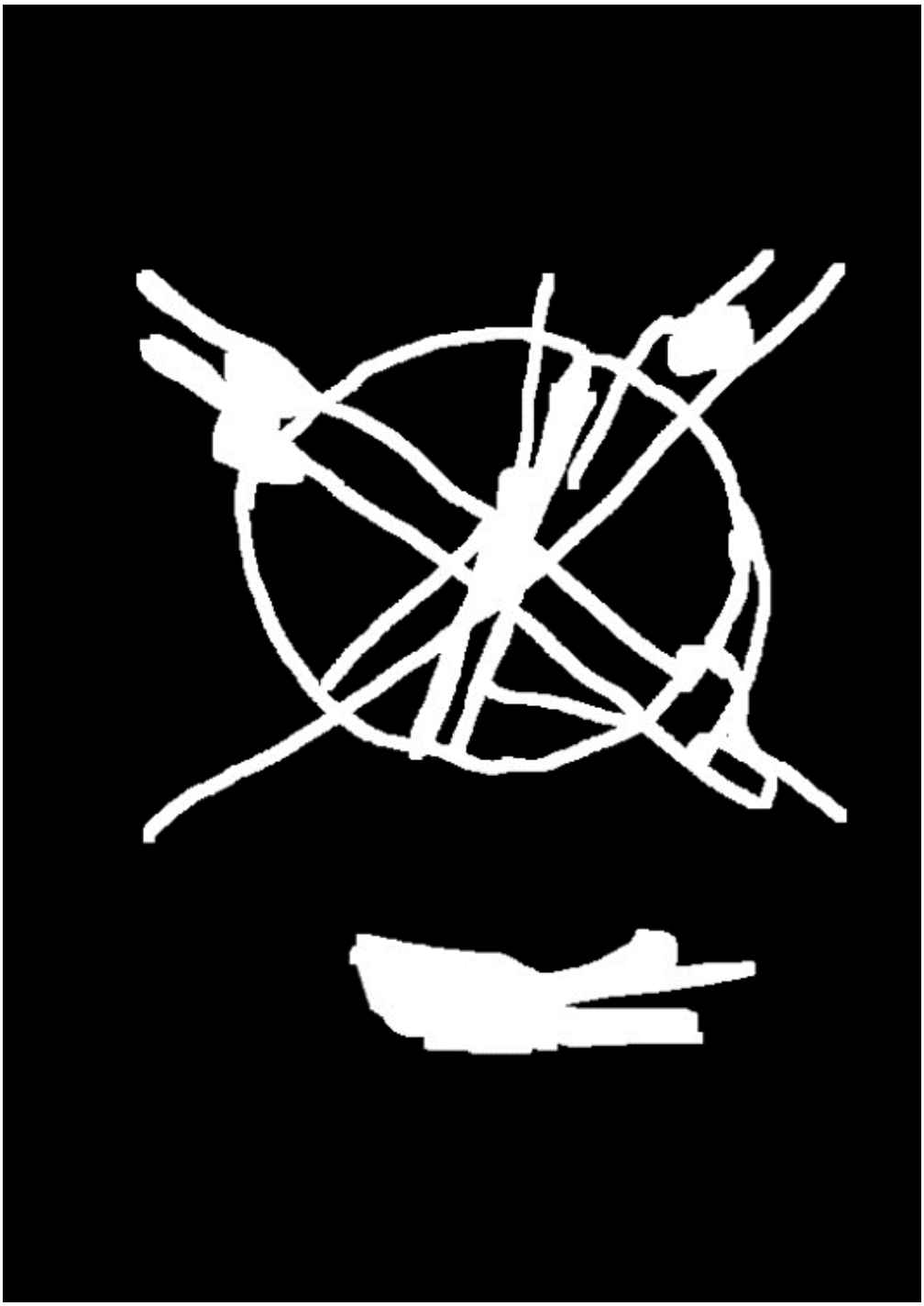

Figure 26: The ground truth regions of difference between Figure 21 and 22

Table 4: AIP Chart Example Evaluation

	Sensitivity	Specificity	Precision	FNR	Jaccard
CNN	0.687304319	0.679911604	0.219240054	0.312695681	0.19935528879694003
SSIM	0.73142715	0.706717824	0.245933755	0.26857285	0.22556429439222456
Pixel to Pixel	0.36625267	0.847525048	0.239038649	0.63374733	0.16909660064521687

As seen in Table 4 in this example, there are fewer correctly identified pixel regions than previous examples, as shown with slightly lower sensitivity scores, and the specificity indicates that most unchanged pixels were also identified as unchanged. So in both cases, the neural network-based feature extraction model performed comparably to the structural similarity-based comparison. However, the slightly higher (but still low) precision does continue to illustrate that many pixels are falsely identified as changed when they are actually unchanged. Here a higher false-negative rate indicates that fewer pixels identified as unchanged were correct and were truly unchanged when compared to the previous examples. A higher portion of changes may be missed here, but the majority are not. The Jaccard similarity scores remain low as the number of false changes identified is high compared to the number of true changes.

5.3 Artificially Generated Example to show shift-invariance

A scenario this model was intended to account for specifically was a chart with only a minor change in values due to small shifts. Here an artificial example was created to illustrate this. Where the same image is produced and shifted by three pixels down and three pixels to the right. The two images are otherwise on the same chart. The structural similarity model and the pixel to pixel comparison generated substantial changes due to the shift. The neural network feature-generated prediction, however, performed substantially better with far fewer changes predicted.

Table 5: Evaluation of an artificially shifted example with no actual changes

	Sensitivity	Specificity	Precision	FNR	Jaccard
CNN	0	0.979889195	0	0	0
SSIM	0	0.755118975	0	0	0
Pixel to Pixel	0	0.766970486	0	0	0

As seen in Table 5, because there are no true-positive examples, there is less that can be said about the actual changes detected. The approach used here is designed such that these minor shifts should not be detected as changes. The specificity score is the score to examine in detail here. This score shows the ratio of correctly identified non-changed pixels over the actual number of non-changed pixels. The high specificity of the CNN model shows the success of the shift-invariance in the model. The majority of unchanged pixels were accurately identified as unchanged. The other metrics are marked as zero here, as there were no truly changed pixel regions.

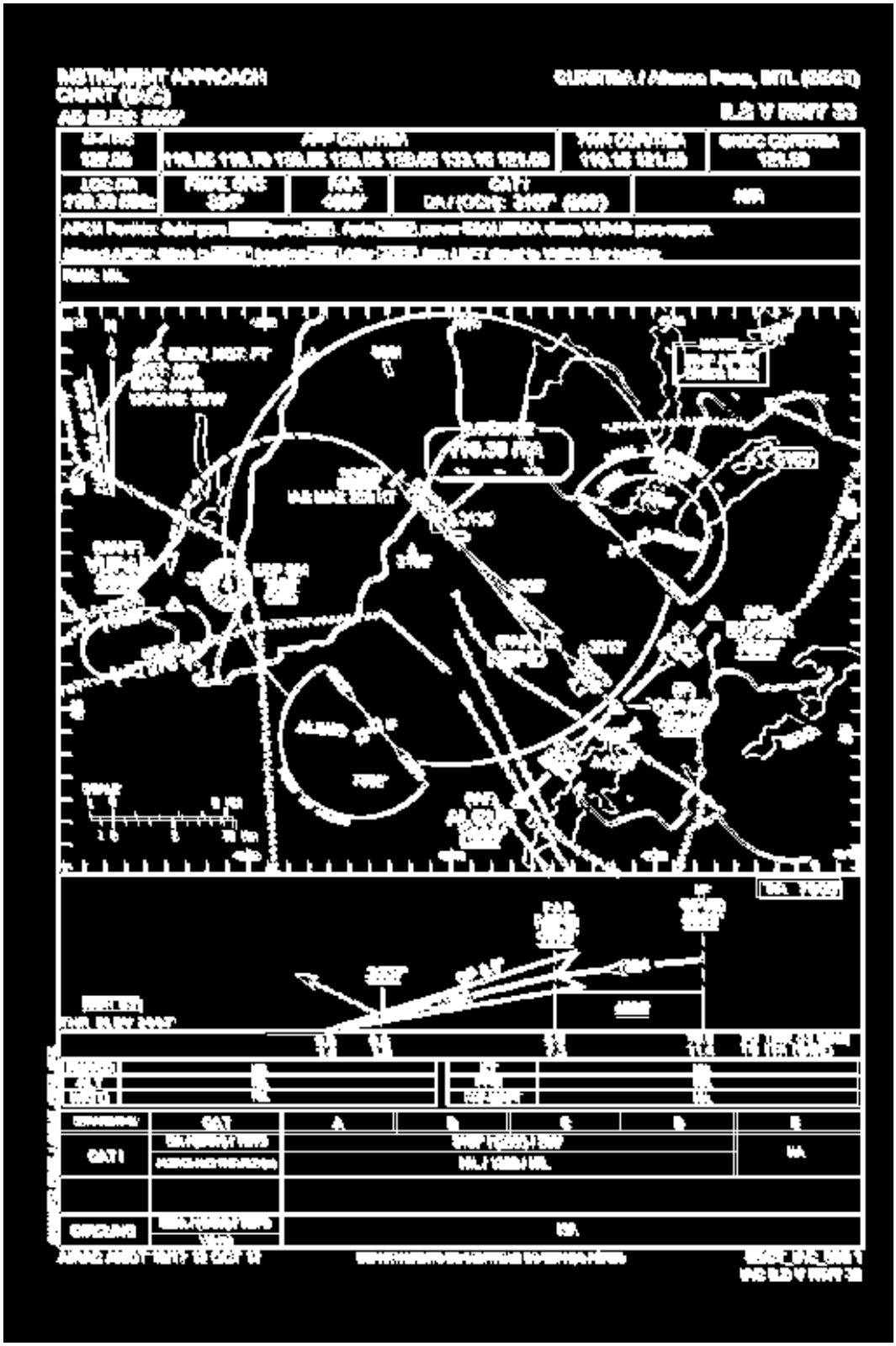

Figure 27: Pixel to Pixel comparison of a shifted copy of an image

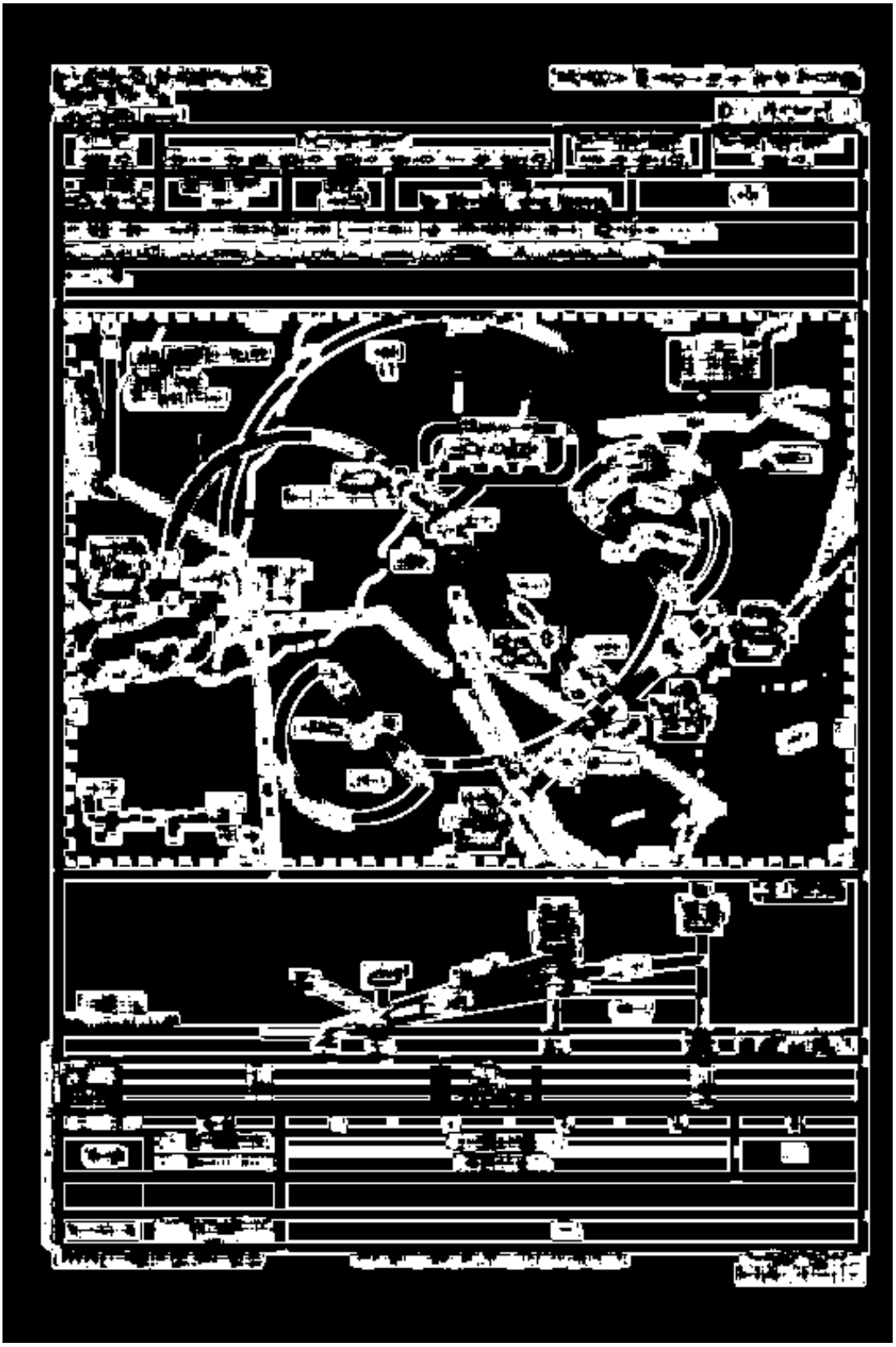

Figure 28: Structural Similarity Threshold-based comparison of a shifted copy of an image

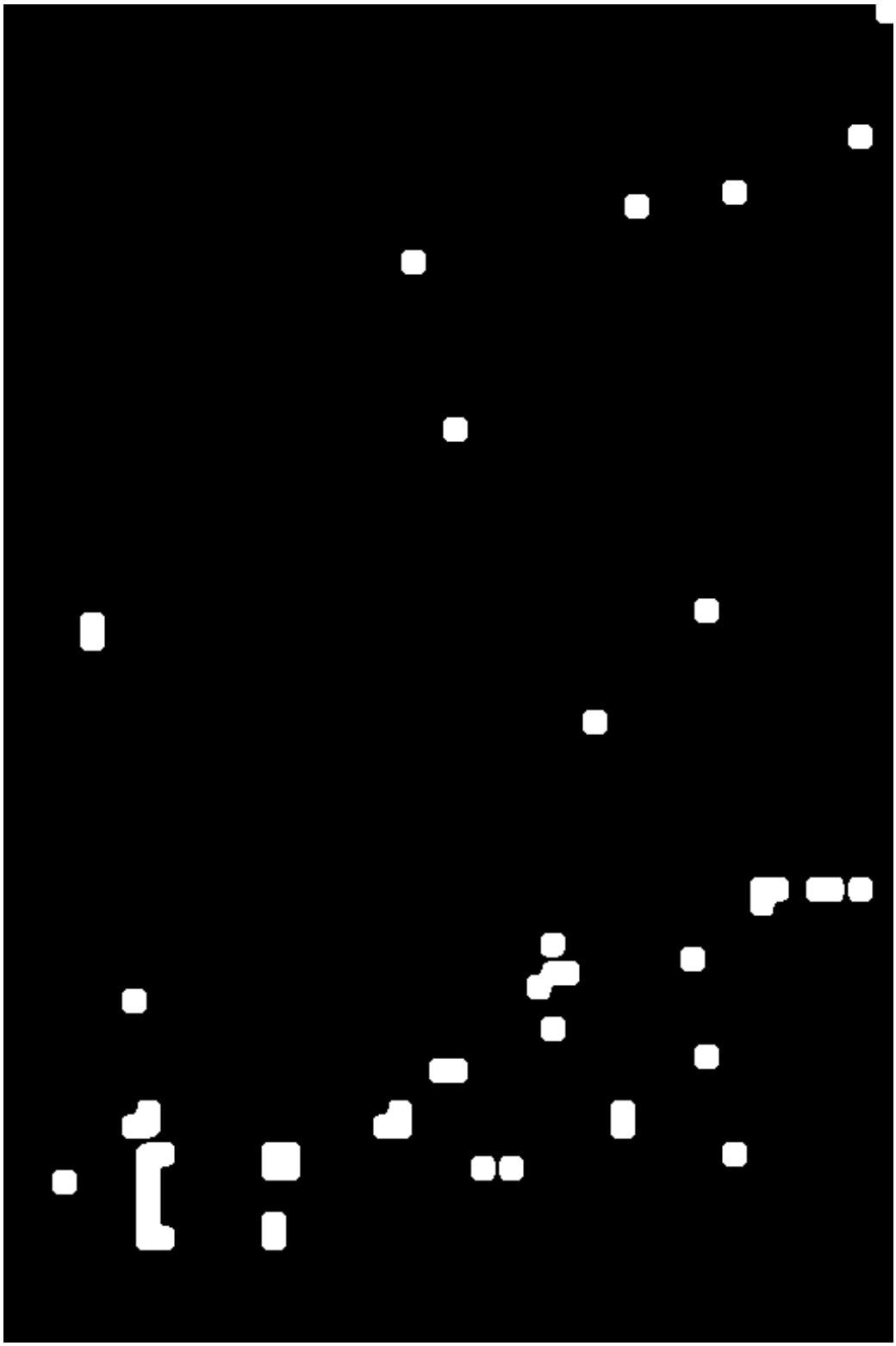

Figure 29: Neural Network based comparison of a shifted copy of an image

5.4 Overall Results

Using all the examples provided, counting each pixel either as changed or unchanged, the results are indicated in the following table. This was done using the fifteen samples of changes produced and hand-annotated. These results indicate that the model does accurately predict most change and changed regions (roughly seventy percent of the time). It also performs well at detecting the negative results as true non-changed regions were also often correctly identified. Precision and the False Negative Rate are the two metrics which performed more poorly. If substantial shifts are not considered changes, then even with some shift-invariance, the models still over-report changes being detected.

Table 6: Overall Evaluation of All Pixel Predictions on All Sets of Changed Images

	Sensitivity	Specificity	Precision	FNR	Jaccard
CNN	0.691051295	0.838249797	0.204392393	0.308948705	0.10774528687328547
SSIM	0.615326057	0.787641655	0.148381982	0.384673943	0.0104377827065099
Pixel to Pixel	0.364472464	0.82778777	0.112895324	0.635527536	0.014174591081388677

Chapter 6

Detecting small differences in images that have been shifted and scaled in difficult to predict fashions is a challenging task. Here a system has been proposed to detect changes, which removes some, but not all, false positives. It does detect changes and is capable of illustrating those changes to a user. While some of these changes may not be meaningful and in line with the ideal, the system detects these changes and can present them to the user to reduce some, but not all, proposed changes. Further work in this topic to account for the types of changes to remove as non-meaningful would require more shift-invariance than this system proposed. Using higher stages than the output of stage 3 of the Mask R-CNN model, for example, would increase the abstractness but further remove the resolution. Often the changes detected at that level were more meaningful but would increase the number of false negatives. Given this project's goal is to predict if there are changes, it is rational to assert that false negatives should be given a higher priority as a thing to prevent than false positives.

It is also worth noting that the substantial difficulty of accurately noting small regions of change and disregarding other regions in the ground truth labeled data has a major impact on the evaluation report. It would be difficult to individually mark each and every pixel accurately when noting which regions should detect changes and which regions should not. There is also a certain level of uncertainty involved. Some ground truth marks may, for example, be close enough to a change such that they were included but were not predicted by any of the models. The small sample size of ground-truth annotated data is also a challenge. While the neural network was able to train on thousands of image examples, there were substantially less samples where there were two versions and noticeable changes. Producing a larger and more accurate dataset will be necessary for further work on this topic. The tools created here, however, do provide a starting point for creating substantially more training data with more samples. Further researchers may use these tools to produce estimations of the ground truth data for more samples, then hand correct the regions so as to create more training data, which could be used for a further revised neural network in the future.

References

[1] L. Shapiro and G. Stockman, Computer Vision, Prentice-Hall, 2001.

[2] W. Shi, M. Zhang, R. Zhang, S. Chen and Z. Zhan, "Change Detection Based on Artificial Intelligence: State-of-the-Art and Challenges," *Remote Sens,* vol. 12, 2020.

[3] R. Radke, S. Andra, O. Al-Kofahi and B. Roysam, "Image change detection algorithms: a systematic survey," *IEEE Transactions on Image Processing,* vol. 14, no. 3, pp. 294-307, 2005.

[4] Z. Wang, A. Bovik, H. Sheikh and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing,* vol. 13, no. 4, pp. 600-612, 2004.

[5] S. Sandhu and A. Agarwal, "Summarizing Videos by Key Frame Extraction Using SSIM and Other Visual Features," in *Proceedings of the Sixth International Conference on Computer and Communication Technology 2015,* 2015.

[6] S. Tareen and Z. Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," in *International Conference on Computing, Mathematics and Engineering Technologies,* Sukkur, 2018.

[7] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *International Conference on Computer Vision,* Barcelona, 2011.

[8] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," in *European Conference on Computer Vision*, 2006.

[9] C. Harris and M. Stephens, "A combined corner and edge detector," in *Fourth Alvey Vision Conference*, 1988.

[10] M. Calonder, V. Lepetit, C. Strecha and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *Proceedings of the 11th European Conference on Computer Vision: Part IV,* Heraklion, 2010.

[11] S. Russell and P. Norvig, Artificial Intelligence A Modern Approach, New Jersey: Pearson Education, 2010.

[12] T. Hastie, R. Tibshirani and J. H. Friedm, The Elements of Statistical Learning, Springer, 2017.

[13] M. Yang, L. Jiao, B. Hou and S. Yang, "Transferred Deep Learning-Based Change Detection in Remote Sensing Images," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 57, pp. 6960-6973, 2019.

[14] J. Zhao, M. Gong, J. Liu and L. Jiao, "Deep learning to classify difference image for image change detection," in *2014 International Joint Conference on Neural Networks (IJCNN)*, Beijing, 2014.

[15] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, pp. 2278-2324, 1998.

[16] S. Liu and W. Deng, “Very deep convolutional neural network based image classification using small training sample size,” in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Kuala Lumpur, 2015.

[17] A. Krizhevsky, I. Sutskever and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Commun. ACM*, vol. 60, p. 84–90, 2017.

[18] Y. LeCun, K. Kavukcuoglu and C. Farabet, “Convolutional networks and applications in vision,” in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010.

[19] V. Nair and G. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines,” in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010.

[20] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Region-Based Convolutional Networks for Accurate Object Detection and Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligenc*, vol. 38, pp. 142-158, 2016.

[21] R. Girshick, “Fast R-CNN,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[22] S. Ren, R. Girshick and K. He, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, p. 1137–1149, 2017.

[23] S. Marsland, Machine Learning An Algorithmic Perspective, CRC Press, 2015.

[24] K. He, G. Gkioxari, P. Dollár and R. Girshick, “Mask R-CNN,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 386-397, 2020.

[25] R. Caye Daudt, B. Le Saux and A. Boulch, “Fully Convolutional Siamese Networks for Change Detection,” in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018.

[26] K. Lim, D. Jin and C.-S. Kim, “Change Detection in High Resolution Satellite Images Using an Ensemble of Convolutional Neural Networks,” in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2018.

[27] R. C. Daudt, B. Le Saux, A. Boulch and Y. Gousseau, “Urban Change Detection for Multispectral Earth Observation Using Convolutional Neural Networks,” in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 2018.

[28] Federal Aviation Administration, *Aeronautical Chart User’s Guide Terminal Procedure Publications*, Aeronautical Information Services, 2020.

[29] K. Wada, *labelme: Image Polygonal Annotation with Python*, https://github.com/wkentaro/labelme, 2016.

[30] K. Wada, A. B. Jung, J. Crall, S. Tanaka, J. Graving, C. Reinders, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, Z. Rui, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer and et al., *imgaug*, https://github.com/aleju/imgaug, 2020.

[31] W. Abdulla, *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*, Github: https://github.com/matterport/Mask_RCNN, 2017.

[32] F. Chollet and et al, *Keras*, https://keras.io, 2015.

[33] M. Abadi, A. Agarwal, P. Barham and et al, *TensorFlow: Large-scale machine learning on heterogeneous systems*, tensorflow.org, 2015.

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar and L. Zitnick, "Microsoft COCO: Common Objects in Context," in *European Conference on Computer Vision*, 2014.

[35] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[36] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Vita

Anthony Marchiafava was born in Metairie, Louisiana. He obtained his Bachelor's degree in Philosophy from the University of Louisiana in 2014. He joined the University of New Orleans as a Computer Science first as a postbaccalaureate student, then as a Master's program student, and worked as a graduate teaching assistant and researcher for Dr. Mahdi Abdelguerfi. He applied machine learning and computer vision techniques to detect changes between aircraft terminal procedures charts as part of his Master of Computer Science thesis.