Leveraging Models of Human Reasoning to Identify EEG Electrodes in Images with Neural Networks

Dr. Alark Joshi
University of San Francisco, USA

Dr. Phan Luu
Philips Neurodiagnostics, USA

Dr. Don Tucker
Philips Neurodiagnostics, USA

Steven Shofner
Philips Neurodiagnostics, USA

ABSTRACT

Humans have very little trouble recognizing discrete objects within a scene while performing the same tasks using classical computer vision techniques can be counterintuitive. Humans, equipped with a visual cortex, perform much of this work below the level of consciousness and, by the time a human is conscious of a visual stimulus, the signal has already been processed by lower order brain regions and segmented into semantic regions.

Convolutional neural networks are modeled loosely on the structure of the human visual cortex and when trained with data produced by human actors are capable of emulating its performance. By black-boxing the low-level image analysis tasks in this way, we can model our solutions to problems in terms of the workflows of expert human operators, leveraging both the work performed pre-consciously and the higher-order algorithmic solutions employed to solve problems.

Keywords: Geodesic Photogrammetry System (GPS), Photogrammetry, Electroencephalogram, Segmentation, Python, TensorFlow, TFLearn, Keras, Deep Learning, Convolutional Neural Networks

INTRODUCTION

Describe the general perspective of the chapter. End by specifically stating the objectives of the chapter. Until very recently, approaches to computer version have been dominated by the application of what the authors refer to as classical computer vision techniques. These include morphological operations, thresholding, feature extractors, Hough transforms, and edge detectors among many others. As researchers and practitioners have gained experience and insights into the problem, slow and steady progress has been realized, but with the advent of the era of deep learning the rate of advancement has been astonishing.

In “The Unreasonable Effectiveness of Data”, the authors argue that “we should stop acting as if our goal is to author extremely elegant theories, and instead embrace complexity and make use

of the best ally we have: the unreasonable effectiveness of data." (Halevy, A. Norvig, P., Pereira, F. 2015). Image processing is to shape analysis and feature extraction what natural languages are to context free grammars, and defining a theory or formula to capture either represents an intractable problem. At the same time, humans are capable of performing these tasks to a reasonable degree of accuracy with little effort. What Halevy, Norvig, and Pereira suggest is that within certain domains no elegant theories can reasonably model the complexity of the problem space, and that data along with machine learning has proven to be an effective means of doing this.

Training a deep convolutional neural network with human labeled data is essentially designing an algorithm capable of emulating the performance of the human labeler(s). In this manuscript, we outline an approach that combines this emulation derived from the labelled data with the algorithmic approach the same users apply in performing a localization and identification task over a set of images.

BACKGROUND

Philips Neurodiagnostics manufactures a complete hardware / software dense array electroencephalography (EEG) systems. EEG, as a technology, has existed since 1875 (Swartz, 1998) and is much older than many other technologies used for non-invasive neural imaging, such as magnetic resonance imaging (MRI) or functional magnetic resonance imaging (fMRI), but offers some unique advantages such as high temporal resolution and movement tolerance.

Epileptic seizures result from excessive synchronization in the firing of neurons in the brain (often starting in a small highly localized region). High temporal resolution makes EEG ideal for the diagnosis of epilepsy, as the associated spikes are most easily characterized by the sub second resolution enabled by EEG. In some cases, epileptic seizures are not controlled by pharmacological means and a determination is made that a brain resection should be performed to remove the tissue that serves as the source of the problem. When resectioning is deemed appropriate source localization of the recovered electrical signals can assist in identifying the region of the brain responsible. In order to perform source localization using the recovered electrical signals, the sensor locations must be registered to the surface of a model of the head.

There are a number of other situations where the ability to localize recovered signals is highly desirable. There is some evidence that neuronal activation can be depressed by low frequency pulsed direct current stimulation [cite]. This may offer a non-invasive means for the treatment of neurological disorders like Epilepsy or Parkinson's. For these disorders, it may be possible to use source localization of signals in EEG to identify the region of origin of the unwanted activity and then, given the reciprocity principle (Fernández-Corazza, Turovets, Luu, Anderson, & Tucker, 2016) (which holds that electrical current will follow the same pathways when injected at the scalp as it follows to reach the scalp from a specific brain region), target the same region with transcranial injected current to modulate the activation or plasticity of the region.

In both cases, an accurate registration of the sensors to the scalp surface is needed. Several means of achieving this exist with their own advantages and disadvantages.

RF systems can derive 3D position by measuring an RF signal emission at a number of receivers. These systems require a technician measure the placement of sensors while the subject remains completely still and is sensitive to the effects caused by other materials in the vicinity of the receivers. Laser scanners can derive very precise models quickly, but pose a risk to eyes and produce a dense mesh that would need to be further analyzed to derive sensor positions. Photogrammetry is a technique that allows the subject to be

imaged quickly, poses no risk to the eyes, and produces a permanent record of net application, but requires advanced image analysis to identify and localize the sensors.

Despite the challenges, the advantages of photogrammetry for this purpose lead to the development of the Geodesic Photogrammetry System (GPS): 11 cameras arranged in a geodesic dome structure are triggered simultaneously and the captured images are analyzed offline to derive the 3D positions of the sensors placed on the head of the patient. Though classical computer vision techniques have been applied to identifying these sensors, significant user intervention is still required to adjust placement and identification of the sensors.

Because of the permanence of this record it is also possible the extract a significant amount of data that can be used to train a neural network to perform the tasks and achieve a higher level of automation.

Neural Networks

Deep neural networks (DNNs) are subset of artificial neural networks (ANNs) with many hidden layers between the input and output. ANNs are modeled on biological neurological systems where neurons form networks such that the neurons have a many to many relationship (the outputs of many neurons connect to the input of a neuron and its outputs connect to the inputs of many more neurons). The neuron possesses an activation function that takes sum of its inputs and produces an output. The inputs are weighted such that some inputs are strongly correlated to the output and others are weakly correlated or not correlated at all.

Artificial neural networks generally organize the neurons into layers where the output of the neurons on one layer feed into the inputs of the neurons on the next layer. Training an ANN is the process of adjusting the weights and biases of the neuronal inputs by providing the network with training data and comparing the output to ground truth until the desired output is achieved.

A well-known result found training even a simple 2 layer, 3 node network to be NP-Complete (Blum & Rivest, 1989). This being so, how is it that the training of deep neural networks can be performed so efficiently? In the paper “On the Computational Efficiency of Training Neural Networks” the authors asked and attempted to answer this exact question finding a few factors that contribute to the efficiency of training modern neural networks: using an activation function that facilitates optimization using one of the gradient descent optimization techniques, over-specification of the networks, and regularization (Livni, Shalev-Shwartz, & Shamir, 2014).

Layers

In an ANN a layer is a collection of neurons that all receive input from the previous layer and all provide output to the next layer. Many kinds of layers exist, some particularly well-suited to specific tasks, like convolutional layers with regards to image processing.

A layer can be configured with an activation function, allowing it to learn to operate on the data flowing through the network by a process of training, or may implement a normalization or regularization function. There are a number of different activation functions, with different characteristics including threshold, sigmoid, tanh, and linear activation functions, the rectified linear unit (ReLU), and many more. The threshold activation function does not produce results that are conducive to optimization by means of stochastic gradient descent (Livni, Shalev-Shwartz, and Shamir, 2014) and is rarely if ever used, while sigmoid and ReLU activation functions are very common in modern architectures.

A convolutional layer, for example, is a layer modeled on the visual cortex and features local receptive fields that take as input submatrices of the previous layers output in order to maintain spatial relationships. Additionally, all of the receptive fields share weights and biases so that the learned image feature is location invariant. As the name suggests, a convolutional layer essentially learns a convolutional kernel that can be applied to an image to recognize a particular type of feature.

Some layers, while still operating on the incoming data, do not possess an activation function or weights that can be adjusted and do not learn from the information that passes through them. An example of this is a max-pooling layer. A max-pooling layer implements something similar to a local receptive field (a component of convolutional layers) that takes inputs from the previous layer and produces an output equal to the maximum value its local receptive field sees. These are commonly used along with convolutional layers and reduce the dimensionality of the output.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a kind of ANN composed of convolutional and, commonly, pooling and fully connected layers. In general, the architecture of a CNN follows some variation of the following: an input layer feeds into some number of convolutional layers that are paired with pooling layers and which feed into a fully connected layer with a number of outputs equal to the number of classes the network is to be trained to identify. Much of the advancement in the state of the art in image analysis in recent years has been built on the back of deep CNNs and many variations on the basic architecture have been tested.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is a competition where research teams develop applications that classify objects and scenes in the ImageNet visual database. The first couple years of this competition the winning teams utilized a traditional approach and achieved ~25% error rate in image classification. In 2012, the winning team utilized a deep convolutional neural network (CNN) and reduced the top 5 classification error to ~16%. Since then, the competition has been dominated by CNNs and the error rates have dropped to a few percentage points – even exceeding human performance.

Currently, Google's Inception v4 network has achieved a 3.07% error rate on the ILSVRC dataset by integrating a number of recent advancements that allow very deep networks to be effectively trained. By some accounts this result is better than human classification which has been reported to be approximately 5.1% (Russakovsky et al., 2015).

BOOTSTRAPPING AN AUTOMATED IMAGE PROCESSING PIPELINE

The advancements in the techniques used to build and train deep neural networks have made it clear that, given sufficient training data, many tasks that seemed impractical only a few years ago are now clearly within the realm of possibility. Furthermore, there are a number of open source libraries and tools that have turned the development and training of deep neural networks into an approachable field, and make experimentation easy.

Possibly the most challenging part of the process is the acquisition of labeled data suitable for training these networks. Tasks that stand to benefit most from the promise of the automation promised by deep learning are often tedious or challenging to perform (this is the very reason the promise of automation is so attractive), thus acquiring sufficient data to use for training presents difficulties.

Once data is collected, a number of further steps and considerations must be taken: How should the training data be presented to the learning algorithm? What kind of network is suitable for the task? How deep does

the network need to be? How many image features should be recognized at each level of abstraction (activation layer)?

Often questions about the number of image features to be identified in each layer, which will affect the design of the network, can be difficult to answer without empirical data, so network design can be an iterative process. Because of the need to train, iterate, and retrain it is important to have access to capable hardware. What might take days on a CPU (as training frequently does), can be performed in minutes or hours on a modern GPU with sufficient memory.

Deriving an Algorithm

Merriam-Webster defines an algorithm as "a step-by-step procedure for solving a problem or accomplishing some end especially by a computer". By abstracting the underlying complexities related to low-level computer vision it is possible to focus on modeling high-level conscious problem solving, and to derive an algorithm by modeling it.

Collecting and Structuring the Training Data

Because significant training data can be required to train a competent neural network, it is often impossible or difficult to automate a task using supervised learning until significant work has been performed to generate a corpus of labeled data. Early wins with deep learning owe their success to a large collection of data produced by hand, such as machine translation where U.N. documents which were produced in many languages were among the sources of training data (Halevy, Norvig, & Pereira, 2009). The core takeaway is that it is possible to begin with a manual task, and if the inputs to that task and the outputs of the task are preserved and can be correlated, it should be possible to extract the necessary information to train a suitably architected network to emulate, to some degree, the work performed by the humans, thus bootstrapping a deep learning approach to the problem.

Designing the Network

In general, deeper networks are more powerful than shallower networks, but costlier in terms of memory and computation. If a network can be executed on a GPU rather than a CPU, the run time can be made orders of magnitude shorter, but will then be constrained by the, usually, smaller pool of memory.

When designing a deep neural network, it is important to consider the problem of exploding or vanishing gradients which can result in difficulties with training. Further, though the performance of any n layer network can theoretically be shown to be at least equal to the performance of an n-1 layer network by configuring the additional layer to simply behave as a pass-through of the preceding layer, in practice performance tends to increase to a certain network depth after which adding more layers degrades it. This is not due to overfitting, but the inability of multiple non-linear layers to learn an identity (pass-through) mapping (He, Zhang, Ren, & Sun, 2015). Residual networks are a recent advancement that have been shown to improve the trainability of deep networks by making it easier to optimize each few stacked layers to the identity mapping.

Determining general aspects of an appropriate network architecture for each task can be accomplished by consideration of the inputs and the outputs. As a rule, convolutional layers work well for image analysis or other tasks with data encoding spatial relationships. Recurrent layers work well for sequential data, so while a convolutional network would be a good choice to perform analysis on the individual frames of a video, a recurrent network would be a good choice for performing analysis on the change over time occurring in the entire sequence. For low dimensional data without spatial or sequential relationships, a fully connected network might be a good choice. However, because of the density of connections (the output of every

neuron in layer n-1 connects to the input of every neuron on layer n), the size of the input data should be minimized to avoid networks that require more compute resources than are available. Furthermore, a simple dense network has no effective means of representing or preserving spatial or sequential information in the input data, and so is not the ideal choice for these tasks.

Many networks will be a combination of these different types of layers along with other layers which may or may not implement an activation function (meaning that they simply perform some arithmetical operation and are not capable of learning). The output layer, for instance, of many convolutional neural networks is a fully connected layer.

Training the Network

Once the data has been organized and the network has been designed, there are a number of variables to consider when deciding how to train the network.

On a macro scale, training consists of two stages: training and validation. During training, the optimizer updates the weights and biases of the neurons in order to minimize the error between the prediction and the ground truth (as described by the labels). During validation, this error is calculated for the validation data but the weights and biases are not updated. The validation step is intended to minimize the degree to which the data is being overfit. Because validation data is never presented to the network while it is being trained, it cannot anticipate what form it will take and if overfitting occurs with the training data, the error computed for the validation data should move inversely to the error during training.

For this reason, the stability of the selection of training versus validation data is essential. So, whatever means is used to segment the training data from the validation data it must perform the segmentation the same way each time. This should be balanced by the need to ensure that biases inherent in the ordering of the data are minimized (For instance, in the case of the application described in this chapter, the order of the extraction of the patches was driven by the numerical identity of the sensors ensuring that these relationships were encoded in the extracted images). A simple solution is to select the inputs for each category using a random function, but specifying an unchanging seed for the function.

Objective Function

The objective functions used during training determines the nature of the error calculation. A categorical cross-entropy objective function, for instance, measures the error in classification for outputs with mutually exclusive classes, whereas a binary cross-entropy objective function measures the classification error of non-mutually exclusive classes. If the task were to identify objects in an image that were members of a single class (either cats or dogs, for example), categorical cross-entropy would produce the desired results, whereas attempting to identify an object that might belong to several classes would suggest binary cross-entropy (cats, dogs, snakes, reptiles, or mammals, where cats and dogs are distinct classes, but also belong in the mammal class).

Optimization Function

The optimization function determines how the training will proceed. Most optimization functions implement some form of gradient descent where the gradient of the objective function's domain is estimated in order to compute the update values that are then backpropagated through the network. Backpropagation is the process by which the contribution to the total error from each neuron is estimated and then used to update those neurons (Hecht-Nielsen, 1988). Plain gradient descent has a number of problems as applied

to the training of deep neural networks, so a number of variants such as Stochastic Gradient Descent (SGD), Momentum, and Adaptive Moment Estimation (Adam) have been proposed and tested.

Stochastic Gradient Descent performs a parameter update for each training sample, foregoing the need to compute the parameters for the entire training set as plain gradient descent, resulting in significant performance improvements and reduced memory requirements. Momentum is a variant of SGD where the concept of momentum is used to constrain the size of the update, addressing the tendency of SGD to oscillate around optima with steep gradients because the optimum value is overshot. Adam, similarly attempts to adapt the learning rate to the local gradient by computing both the momentum and an exponentially decaying average of all past gradients.

Convergence

Convergence is the goal of training, where the loss is minimized and further training does not improve performance. Depending on the architecture and complexity of the network and the nature of the data it may take many epochs (or iterations through the training data) to converge or just a few. At the completion of an epoch, a loss/accuracy vs. validation data is typically performed. The average of the change trajectory of these metrics will asymptotically approach zero as the network approaches convergence.

Non-optimal convergence can be symptomatic of one of a few problems: The training data may be too noisy, represented by a form inappropriate for training, or insufficient. The network can be too shallow to represent the desired mappings. There might be issues with class imbalance, or with class definition.

Memory Consumption/Mini Batch Training

With the advent of deeper networks and large training sets it is commonly impossible to maintain the network and training data in memory and becomes essential to train the network using an approach commonly referred to as mini batch training, where a small subset of the training data is provided to the network at once.

MODELING THE HUMAN APPROACH

Human experts frequently perform image analysis tasks that we don't currently have automated solutions to. Normally these experts have developed an internal algorithm used to solve the problem, but which is not directly transferrable to computer automation because this algorithm relies on a number of non-linear and pre-conscious steps that the operator has no intuition about. If this approach to the problem can be specified such that the non-linearities are partitioned to discrete steps that function in the context of a deterministic algorithm, the inputs to and outputs from the non-linear functions can be defined. If the inputs and outputs are known, it should be possible to collect labeled examples of such, design a network appropriate to problem, train it, and drop it into the algorithm as long as the complexity of the task is not too great to be modelled by the network.

Deriving an Algorithm

In the case of identifying and locating EEG sensors in the images, users have to overcome a number of challenges that have frustrated current approaches to automating the task.

The first stage in the location and identification of sensors is segmenting the image into foreground (sensors) and background (everything else). Once the images have been segmented into foreground elements and background elements, the sensors in the foreground are examined individually in order to determine the unique identity. There are, then, usually sensors that cannot be identified by examining the label as it might be overexposed or obscured by hair, wires, or ears. For these situations users are provided with a map of

the positions of sensors they use to determine the identity of an unidentifiable sensor through an examination of nearby sensors.

From these stages, a high-level algorithm was derived. Figure 1 is an expression of this algorithm specified in Python.

Figure 1. The derived algorithm specified in Python

Algorithm 1: *Identify Sensors*

```
def identify_sensors (image):
  segmented_image = segment(image)
  sensor_locations = find_centroids(segmented_image)
  results = []
  unidentified_sensors = []
  For sensor_location in segmented_image:
    sensor_identity = identify_sensor(sensor_location)
    If sensor_identity != unknown:
      results.append((sensor_identity, sensor_location))
    else:
      unidentified_sensors.append(sensor_location)
  For sensor_location in unidentified_sensors:
    predictions = {}
    for known_sensor in Results.nearest_sensors(sensor_location, 6):
      prediction, response = identity_given_displacement(camera, known_sensor, sensor_location)
      If response > response_threshold:
        try:
          predicted_identity[prediction] += response
        except KeyError:
          predicted_id[identity] = response
      predicted_identity = sorted(predicted_id, key=predicted_id.get, reverse=True)[0]
      if predicted_id > identification_threshold:
        results.append(predicted_identity, sensor_location)
  return results
```

The bolded function calls represent steps that are complex operations and well suited to the application of deep learning, while the rest of the algorithm consists of fairly simple applications of Boolean logic or geometry. The `segment(image)` function performs ***image segmentation***, the function `identify_sensor(sensor_location)` performs ***visual sensor identification***, and the `identity_given_displacement(camera, known_sensor, sensor_location)` function performs ***geometric sensor identification***.

COLLECTING AND STRUCTURING THE TRAINING DATA

The derived algorithm defines a solution to the problem of labeling the sensors in an image, but relies on three undefined functions. These functions could be fulfilled by human effort, classical computer vision, or deep learning approaches, but framing the problem in this fashion exposes the nature of the problem in a way that exposes the aspects of the problem that can benefit from recent advancements in machine learning.

In the case of the GPS, the output of the work of users is a set of files containing the 2D coordinates of the sensors within the individual camera images as verified by the users, the 3D coordinates derived using the techniques of photogrammetry, and the images captured by the cameras. Using these three pieces of information, it was possible to extract a set of data that was used to train the neural networks to recognize sensors in the images with a high degree of accuracy.

For the problem of labeling sensor positions in images, a software application has been in use for some time. While the operation of the application is imperfect and requires a not insignificant amount of user effort, its outputs can be used to train a neural network to perform the undefined functions.

An examination of the inputs to and outputs from the undefined functions reveals what information is required for each.

Image Segmentation

The input to the segmentation function is a grayscale image produced by one of the cameras, while the expected output of the function is a binary image with sensor positions highlighted. This suggests that a neural network be trained to classify each pixel in the image into one of two classes: foreground (sensor) and background (everything else).

From each of the images a set of patches centered around the sensor locations and a second set of patches centered at random points not containing sensors were extracted. The network was trained with these two classes to discriminate between sensors and background texture. Once the network was trained, a sliding window would be moved over the image, classifying each pixel as belonging to a sensor or to the background and yielding the segmented image as required.

One important consideration when collecting the data is class balance. If there is an imbalance in the size of a class, the network will learn to favor the larger class, especially in indistinct cases. This can be desirable, but should be considered. In the case of the images the image segmentation network is to operate on, there is significantly more background information than foreground, so the size of the non-sensor class was expanded to approximately 4 times the size of the sensor class.

Visual Sensor Identification

The input to the visual sensor identification function is the location of a of a sensor in the image. This information comes from the segmented image (determine the centroid of each region marked as containing a sensor by the segmentation step). The output of this function is the identity of the sensor as determined by an examination of the label.

The nets used for this research work contained 258 unique electrodes (256 EEG sensors, one common, and one reference), suggesting that the network should be trained to classify the input data into one of 258 classes. To allow this network to eliminate false positives from the segmentation a 259th class was added to again represent background noise.

For each of the images in the training data, patches were again extracted, but this time they were separated into the 259 classes described. The network trained on this information was then presented with image patches extracted from an image at the points determined by the output of the segmentation function.

Geometric Sensor Identification

The input to the geometric sensor identification function is the camera number, the 2D location of an unidentified sensor and the 2D location of a known sensor. A displacement vector between these sensors is computed, and based on the known geometric relationships between sensors, an identity is predicted for the unknown sensor, modelling the strategy applied by users to the identification of sensors with unreadable labels.

The solved files were, again the source for this information. For each recorded user verified 2D sensor location, it and the nearest 6 sensor locations were extracted. For each of the nearby sensors, the displacement vector to the sensor in question was provided as input, and the identity of the unknown sensor was provided as the label.

DESIGNING THE NETWORK(S)

In the algorithm described, three functions were laid out that were appropriate for the application of deep learning solutions. An understanding of the input data, the operations being emulated, and the output data suggested the appropriate network architecture for each. As some networks preserve spatial relationships while others preserve sequential relationships, understanding the nature of the problem and the operation of the network is essential to attaining good results.

Image Segmentation

As formulated, performance of this task is ideally suited to an emerging variation on the standard CNN: a fully convolutional network (FCN). Current results suggest that these offer a high performance means of performing semantic segmentation of arbitrarily sized images (Shelhamer, Long, & Darrell, 2016).

Instead, the authors implemented this step using a normal convolutional network and reformulated segmentation as a classification task. The entire image is covered by a sliding window that computes, for every pixel it examines, a class: either the pixel is part of a sensor or not.

This is a computationally expensive task, requiring about 6 hours per image on a CPU or about 5 minutes using a NVidia GTX 1080ti GPU. Improvements to the computational run time could be obtained by down sampling the input image, but selecting an ideal network architecture (in this case, an FCN) could potentially achieve similar results to the current implementation at full resolution with a sub-second runtime.

Visual Sensor Identification

The input to the visual sensor identification function is an image patch extracted from the original image at the location determined by an examination of the segmented image. The output of this function is a classification of the patch into one of 259 categories. Current results with image classification demonstrate the suitability of CNNs to this class of problems.

The traditional architecture of CNNs consists of an input layer followed by a number of convolutional layers paired with max pooling layers, feeding into one or more fully connected layers with output dimensionality equivalent to the number of classes the network is to be trained for (Krizhevsky, Sutskever, & Hinton, 2012).

Geometric Sensor Identification

The input to this function is 1D vector of length 272. A one-hot encoding is used to specify the camera number (the first 11 elements), the known sensor id (the next 259 elements), and the displacement vector (the final 2 elements). For any unidentified sensor, the six nearest sensors are selected, and for each a displacement vector from it to the unidentified sensor is computed. This is fed to the network, which will make a prediction. The predictions from these tests are, then, counted and the winning identification is assigned to the unknown sensor.

As there are no spatial relationships within the data to be preserved and as the input vectors are relatively small, a fully connected network was deemed appropriate.

REFINING THE NETWORKS

A careful analysis of the results of training the networks suggested improvements to the network architecture and helped to identify problems with the training data.

Image Segmentation

As the patches used for the training of the segmentation network are small grayscale images, the architecture of the network model used to perform the operation was based on a simple convolutional network that was capable of achieving good results on the (similarly small grayscale) MNIST database of handwritten digits. This produced reasonable results, but because the goal of the segmentation stage was to segment photographic images it became clear that a deeper network capable of capturing more abstract image features was necessary. Due to the challenges with training deeper networks, this also meant changing the type of layers. Very good results have been achieved with the training of deep Residual Networks (He et al., 2015). so the network for segmentation was redesigned with many more layers, most of which were residual layers.

In looking at the data, it also became clear that there were more than two classes. The non-sensor patches could be divided into a discrete set of expected elements (wires, bird bands - used in wire management, net structural elements, as well as features of the subject: eyes, ears, nose, etc.) and the sensors were best described as belonging to one of two classes normal sensors and cardinal sensors. This distinction is important because the white/black of the labels of the two classes is inverted.

No data existed capable of automatically classifying background (non-sensor) features, but it was possible to separate normal and cardinal sensors. After extracting data with the sensors separated into these two classes the network was adjusted to output three classes (normal sensor, cardinal sensor, non-sensor) instead of two and retrained.

Visual Sensor Identification

Because the development of the sensor visual identification stage benefitted from the development of the segmentation stage very few iterative changes were made to the design of the network, and is essentially identical to the design of the segmentation stage network with the exception that the number of classes in the output is 259 rather than three.

Refining the Training Data

The extracted patches consisted of about 21,000 images of sensors and nearly 80,000 examples of background texture, too much data to individually verify. Interestingly, after training the network, it was possible to use it to predict the classes of the training data patches. This result allowed the authors to search the images and remove the misclassified patches. If an image was part of the training data for the sensor class, but the network failed to predict this class it was highlighted. The same operation was performed over the non-sensor patches and a user went through the ~2000 patches this operation returned to verify the performance of the search. The accuracy of this search was about 50% in that nearly 1000 of the selected patches were returned to their category of origin after examination. This high error was intentional and due to the decision to relax the classification thresholds for this operation: the authors were not concerned that

some positive results might be returned, but wanted to avoid negative results (where a misclassified patch was left in the wrong class).

With the networks trained, the algorithm was run against separate input that had not been a part of the training or validation data. This highlighted a few deficiencies in the networks. Specifically, during segmentation, cardinal sensors, overexposed sensors, sensors partially obscured by ears, and sensors behind wires were regularly missed. The reason for these issues was related to class imbalance: though there were a number of examples of these types in the training data there were few enough of them that the penalty for ignoring them entirely was small enough that the network didn't bother to learn them. To address this issue, a number of examples of each of these types were duplicated, with no change made to them at all, and the network was retrained.

During the sensor identification stage, it became clear that a number of individual sensors, though clearly in the segmented data were identified as background noise. An examination of the training data for these sensors made it obvious that this was again an issue of class imbalance. In some cases, there were as few as 32 example patches for a given sensor (while most were in the range of 150 - 200). These were duplicated to improve the balance and ensure that the network could learn to identify them.

The results obtained during the geometric identification stage illustrated the importance of understanding the data and considering how the network will interpret it. It was initially impossible to train this network to achieve anything approaching useful results with a validation accuracy never exceeding 50%. This was due to the decision to represent the displacement vector as distance and angle causing vectors in the first 90 degrees and the last 90 degrees to look very different to the network despite being very similar in many cases. After changing this vector representation to use normalized x and y displacement values the network was quite easily trained to a validation accuracy of about 98%.

Issues, Controversies, Problems

The implementation of the technique described in this manuscript was capable of very nearly completely automating the process of locating the sensors within the images as of writing given a short development time and limited data. This being the case, there were still a number of deficiencies the authors identified.

The sliding window approach taken to compute the segmentation of the image is an inefficient means of performing this operation. Newer techniques like fully convolutional networks can perform the same task far more efficiently and are a better fit for the work that is being done.

Due to the sequential nature of the solution, where the output of one step acts as the input of the next, any errors that occur early on in the process are exaggerated by later stages. By design, sensors not properly located during the segmentation stage could not be added by the identification stage, and these "holes" in the sensor net could lead to the geometric sensor identification to misidentify sensors. Because this step is performed recursively in order to build out a full solution, a single misidentification could lead to several more.

The nets contain several items that resemble sensors when overexposed; in particular, the net is populated with a few unlabeled pedestals similar in shape and size to the sensors. In the images these appear as nearly pure white. Additionally, there are a number of "bird bands" used for wire management the, depending on their angle versus the camera can appear as small white circles, similar to the central dots of the cardinal sensors.

Another challenge resulted from class imbalances as described previously. Within the nets there are a few "cardinal" sensors whose labels are inverted (a black donut around a white circle, rather than a white donut around a black circle), and even when clearly visible, they were identified at a much lower rate than the non-cardinal sensors.

SOLUTIONS AND RECOMMENDATIONS

Tools like TensorFlow and Keras (or TFLearn) have made deep learning very approachable, but there are still a number of complexities and details that need to be taken into account.

Plan Ahead

One of the roadblocks to the adoption of deep learning systems is the lack of appropriate training data. Oftentimes, the output of non-automated or semi-automated approaches can be used as training data, but considering the formulation of a deep learning approach to a problem ahead of time can improve the quality or quantity of data or reduce the pre-processing work required. In the case of the work described herein, the semi-automated solution used to generate the training data did not initially capture the user specified 2D sensor positions. Though the 2D positions of the sensors could be recovered by projecting the 3D coordinates into the images, the nature of the problem meant that the 2D projection of a sensor into any image might be slightly displaced (due to the uncertainty associated with finding the intersection of two rays in a 3D space) and requires extra work.

Mind the Learning Rate

The learning rate is an important hyperparameter. A learning rate set too low can negatively impact the time required to train the network, but too high, and the optimizer will overshoot the optima, preventing the network from being trained to the accuracy it would otherwise be capable of demonstrating. If a network converges prematurely this can be a sign that the learning rate is set too high. Before re-examining the training data or the network architecture, lowering the learning rate can be an effective means of achieving (usually) small increases in accuracy performance.

Overfitting

If training accuracy is very high and stable and validation stability is similarly stable, but lower, this can indicate that the network has been trained to recognize the features of the training set, but that there is variance in the data for which it has not been trained. The network is overfitting the training data in this case. If this is the case, a solution would be to provide more training data better capable of capturing this variance.

Understand the Data to Formulate an Efficient and Effective Solution

It is important to consider the characteristics of the data and the desired operation in order to determine the appropriate network architecture. The segmentation step was formulated as a classification task for a small image patch, outputting a class that was assigned to the pixel at the center of the image. This approach worked, but was slow and memory intensive. Where it slid a window over the whole image with a stride of one pixel, an FCN could examine the entire image by dividing it into just a few overlapping patches.

On the other hand, by selecting only a few points in the images, and utilizing a network well-suited to the problem (A deep Residual CNN), the visual identification step performs its work efficiently.

Class Balance

Classes should be balanced in such a way as to produce the desired results. Underrepresented classes will often not impose enough of a penalty during training to ensure that the network learns to identify them. In the absence of more training data to fill out these classes, it can be effective to duplicate the data on hand to ensure that the class will be learned. Doing this will, of course, suffer from a lack of robustness to all of the possible variations that exist.

Consider the cost of a false positive result versus a false negative. If one is preferable to the other, class balances could be adjusted to prefer one over the other.

Understand the Data and How the Network Interprets It

When poor results are obtained, the training data is commonly the problem and understanding the form of the data and how it will be interpreted during training is absolutely essential. As previously described, when training the geometric sensor identification network, the displacement vector was originally specified as an unnormalized vector of angle and distance. By reworking the vector representation to use a simple normalized x-displacement, y-displacement format, the results were quickly improved and the training converged at about 95% versus the validation data set.

FUTURE RESEARCH DIRECTIONS

The authors have attempted to identify as many deficiencies as possible, and have devised approaches to address these deficiencies.

The computational cost of the segmentation step is unnecessarily significant. Fully convolutional networks are defined for exactly this task and have much better computational performance characteristics (Shelhamer, Long, & Darrell, 2016). Given the 11 images and approximately five-minute computational time (on a GPU) for the current approach, deriving a solution with would take nearly an hour, almost all of it for computing the dense segmentation.

The current approach is, in part, a workaround for a lack of sufficient training data. If sufficient data existed, the segmentation step could be extended to segment directly to sensor identity instead of breaking this into two steps. When considering all sensors as being of the same class, the solved files contained at least 21000 examples, but for each sensor, there were generally less than 200 examples. Once the computational performance of the segmentation step has been improved, the current implementation can be used to drastically reduce the amount of time required to compute solutions to these files; at this time more data can be acquired allowing a segmentation step to be trained that can segment the images by sensor identity, reducing the depth of the pipeline and the potential for an early failure to be exaggerated by a later step. In regards to the sensor misidentification problem in the output of the geometric identification stage in particular, utilizing the predicted sensor identities as suggestions and then testing the suggestion by taking the RMS error of the triangulation of the sensor given the suggestion might significantly improve these results by highlighting results that exhibit high triangulation error (which typically indicates a sensor misidentification).

The issue with overexposed sensors being mistaken for extraneous net structures is thorny. It is almost always possible to determine by careful examination of the image which is which, so training a network to differentiate the two should not be impossible. Furthermore, there is a lot of geometric information and inter-image correlations that can also be used to differentiate the two. If the segmentation step selects these as likely sensors, a filtering step could be implemented that would utilize this information to sort them into the appropriate class.

CONCLUSION

Many problems arising in the domain of image analysis can be formulated in terms of algorithms that rely on neural networks trained on human effort to solve non-linearities in the input data. This is, the authors argue, the way humans solve these problems; relying upon pre-conscious regions of the brain to perform operations that feel effortless while devising a set of higher order strategies to operate on the outputs of the pre-conscious work the brain has performed.

While translating directly from input to desired output is ideal, the approach described here is capable of reducing the problem scope to the level of human intuition and facilitates designing approaches to these problems that draw upon the insights of subject-matter experts.

The paucity of training data in some more specialized domains is the biggest gating factor to the implementation of deep learning approaches to these problems. The authors have attempted to show, here, how scarce training data can be used to train a neural network to simulate the human visual cortex in the performance of a task, while using the conscious intuitive approach employed by human volunteers to define a deterministic algorithm capable of performing tasks that were previously difficult to automate. In so doing, the amount of available training data can be rapidly increased (as much of the work of producing this data can then be automated) opening the door to the possibility of designing and training a network capable of moving from input to desired output without the requirement of additional machinery.

REFERENCES

Blum, A., & Rivest, R. L. (1989). Training a 3-node neural network is NP-complete. In *Advances in neural information processing systems* (pp. 494-501).

Fernández-Corazza, M., Turovets, S., Luu, P., Anderson, E., & Tucker, D. (2016). Transcranial electrical neuromodulation based on the reciprocity principle. *Frontiers in psychiatry*, 7.

Halevy, A., Norvig, P., & Pereira, F. (2009). The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2), 8-12.

He, K., Zhang, X., Ren S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. Retrieved from https://arxiv.org/abs/1512.03385

Hecht-Nielsen, R. (1988). Theory of the backpropagation neural network. *Neural Networks*, *1*(Supplement-1), 445-448.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Livni, R., Shalev-Shwartz, S., & Shamir, O. (2014). *On the Computational Efficiency of Training Neural Networks*. Retrieved from https://arxiv.org/abs/1410.1141

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M, Berg, A., & Fei-Fei, L. (2015) *ImageNet Large Scale Visual Recognition Challenge.* Retrieved from https://arxiv.org/abs/1409.0575

Shelhamer, E., Long, J., & Darrell, T. (2016). *Fully Convolutional Networks for Semantic Segmentation*. Retrieved from https://arxiv.org/abs/1605.06211

Swartz, B. E. (1998). The advantages of digital over analog recording techniques. *Electroencephalography and clinical neurophysiology*, 106(2), 113-117.

ADDITIONAL READING

He, K., Zhang, X., Ren, S., & Sun, J. (2016, October). Identity mappings in deep residual networks. In *European Conference on Computer Vision* (pp. 630-645). Springer International Publishing.

Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning* (pp. 448-456).

Jegou, S., Drozdzal, M., Vazquez, D., Romero, A., & Bengio, Y. (2016). *The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation.* Retrieved from https://arxiv.org/abs/1611.09326

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

Simoyan, K & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition.* Retrieved from https://arxiv.org/abs/1409.1556

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *AAAI* (pp. 4278-4284).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Xie, S., Girshick, R., Dollar, P., Tu, Z., & He, K. (2016). *Aggregated Residual Transformations for Deep Neural Networks.* Retrieved from https://arxiv.org/abs/1611.05431

KEY TERMS AND DEFINITIONS

Backpropagation: Process of estimating the part of the total error attributable to each neuron and updating the weights to minimize that error.

Convolutional Neural Network: A neural network consisting (at least partially) of convolutional layers, which learn convolutions that can be applied to an image (or other data) to extract features.

EEG (Electroencephalogram): Non-invasive EEG recovers electrical signals generated by the firing of neurons using sensors at the surface of the scalp, while with intracranial EEG recovers those signals using sensors emplaced in actual brain tissue.

GPS (Geodesic Photogrammetry System): A system consisting of 11 cameras arranged in a geodesic dome structure mounted on a gantry which is used to derive the 3D positions of EEG sensor nets as applied to the heads of subjects.

Photogrammetry: The technique of recovering 3D information through the process of 2D image analysis.

ReLU: Rectified linear unit. The non-linear activation function `A(x) = max(0, x)` with a range of $[0, \infty)$.

Threshold Activation Function: The function `A(x) = 1 if x > 0 else 0` with outputs of either 1 or 0. Because of this, the derivative of the threshold activation function is zero almost everywhere and is not a good candidate for optimization using stochastic gradient descent.