NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Evolving Synthetic Data Generating Programs for Text Recognition Tasks

Evolving a Diverse Population of Synthetic Data Generators using Genetic Programming with Novelty Search for Robust Text Recognition Models

Daniel Cooper

Dissertation presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

EVOLVING SYNTHETIC DATA GENERATING PROGRAMS FOR TEXT RECOGNITION TASKS

by

Daniel Cooper

Dissertation presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

Advisor / Co Advisor: Leonardo Vanneschi

November 2020

In memory of William Geffen, who told me to never stop learning – no matter what

Acknowledgements

I would like to thank the incredible professors I have had the privilege to learn from during the course of this degree. I would like to especially thank Leonardo Vanneschi for his truly excellent teaching style and for introducing me to the fascinating world of evolutionary computation.

Thank you to all my wonderful classmates and the community that arose during the course of study. This was one of the most amazing aspects of study, finding and learning from all the tremendous people along the way.

Finally, thank you to my dear friends and family for all the support along the way.

Abstract

The quality of models produced via supervised machine learning depends on both the learning algorithm used and the training data available to learn from. The work presented in this paper focuses on optimizing training data directly and compares different methods for generating synthetic training data while holding the learning algorithm constant. In this paper, the author proposes a new algorithm that leverages genetic programming to create a diverse population of data generating programs which are sampled from to create training data for the given task. This is applied within the context of building a robust text recognition model that can be integrated into a broader document processing software solution that supports multiple domains.

Keywords: Deep Learning, Novelty Search, Genetic Programming, Synthetic Data, Algorithms, Computer Vision, Document Processing, Document Understanding

Contents

Chapter 1

Introduction

Computer programs produced by machine learning algorithms in a supervised learning context have dependencies on both the algorithm's implementation in computer code and the training data that the program learned from. Making changes to either the learning algorithm or the training data can have a significant impact on the quality of the learned solution for a given task. Much of the published research related to machine learning focuses on new algorithms or modifications to existing algorithms, while holding the training data constant. This provides a quantitative mechanism to compare different approaches and an objective measure of model performance against common benchmarks, but could potentially understate the importance of training data for machine learning.

The goal of this research paper is to provide a simple case study examining the effects of a contrarian approach for supervised learning – holding the machine learning algorithm constant and focusing purely on the training data it learns from. This case study focuses on the highly constrained use case of generating synthetic data for text recognition models to learn from within the domain of document processing. The author examines different methods to generate synthetic training data for the given task, and compares the robustness of the corresponding learned models across various types of documents.

Supervised learning requires training data to be labeled with ground truth annotations, a process that can be time consuming and expensive. This is especially true with modern deep learning algorithms that tend to require large volumes of labeled data. Synthetic data has emerged as one potential solution to reducing the cost of labeling data in the era of deep learning. This methodology involves using a computer program to generate training data where the ground truth is automatically labeled during data synthesis, rather than requiring a human annotator to make manual judgements

on real data that has been collected. Synthetic data is not commonly used for most machine learning projects, but text recognition has been one of the exceptions.

Text recognition is the process of converting an image of text into a machine readable string of text. Machine learning based approaches are the dominant paradigm for producing computer programs capable of text recognition, and most models are trained on synthetic data. The most common approach to generating synthetic data for this task involves hand coding a single data generating program which produces both an image containing text and a labeled text string. This program is then sampled from to produce training data.

In this research, the author explores generating an ensemble of synthetic data generating programs using machine learning rather than hand coding a single parameterized script. The individual data generating programs all produce different types of images (training data) given the same input. The ensemble of data generating programs is evolved using genetic programming, and optimizes for diversity of the training data that is generated. Finally, text recognition algorithms are instantiated and trained on the synthetic data generated by these various approaches, and the resulting models are compared across different document types to test for robustness. The best performing models are then integrated into a document processing and information extraction workflow based on their robustness and overall performance.

Chapter 2

Machine Learning for Documents

2.1 Document Processing and Information Extraction

Manual document processing and information extraction from digital files or scanned images is an extremely common, time consuming, and expensive component of business workflows across multiple industries and business units. For example, almost every business must extract details from inbound invoices and bills as part of their accounts payable process, parse resume details during talent acquisition, and read submitted receipts for expense reporting and management. There are some existing specialized software solutions that attempt to automate manual processing for specific high volume document categories, but in the authors experience they tend not to perform at the acceptable levels required to fully automate their corresponding business processes. These solutions are typically based on hand-written rules / heuristics applied to specific templates, and often fail to generalize robustly to real world data distributions.

In addition to the most common types of documents, there are many niche document categories and a wide variety of specialized business processes for dealing with them. Examples of these include government forms related to small business loans that must be processed in time to provide critical infusions of capital, disclosure documents with information on specialized securities that trading teams must swiftly ingest and make decisions on, and many more. Not only must document processing software deal with the long tail distribution challenge for these niche document categories and corresponding business processes, but new types of documents emerge organically over time (e.g. new tax forms, paycheck protection programs, etc.) and existing document categories change over time (i.e. resumes are becoming for artistic and visually focused[11]). Finally, there is often a one to many relationship between

document categories and tailored business workflows across different industries, business units, and stakeholder groups depending on the business context, so it is very challenging to provide generic solutions.

2.2 Commercial Solutions

Google[12], Microsoft[10], and Amazon[1] all offer machine learning based commercial solutions for generic document processing that includes features such as text extraction, table recognition, and identifying key/value pairs. These commercial services are impressive and have the potential to deliver significant business value, but have some major limitations related to document processing workflows. The first challenge is that they can lack robustness for basic text recognition and optical character recognition – the focus of this thesis. The author experienced this when attempting to build solutions for invoice and resume parsing with the commercial tools available, and believes this may be due to the high levels of noise in the case of invoices and nontraditional colors / font choices in the case of resumes. This has been gradually improving over time, but accuracy requirements can prevent these solutions from being used based on this limitation alone. The second and more significant challenge is that document processing tasks typically required a series of predictions that extend well beyond basic text recognition. For example, in the case of invoice processing, the document must be converted into a semi-structured representation with specific fields such as the due date and total amount due tagged, extracted, and normalized.

These commercial solutions seem particularly limiting because it is highly likely that the representations learned by large scale document processing models will be effective at information extraction tasks[38], and that transfer learning opportunities exist. Current solutions understandably do not make models publically available, nor do they offer an interface to fine-tune models on target domains / downstream tasks – thereby potentially limiting their effectiveness in providing an end-to-end solution. This also speaks to the broader challenge around how to best represent documents in a manner that effectively combines their visual features, semantic and syntactic textual attributes, and spatial relationships between elements and their neighbors for downstream tasks and transfer learning.

2.3 Opportunity Landscape

Machine learning based methods for automatic document processing and information extraction seem poised to disrupt the broader document processing software industry, particularly given the recent success of deep learning based approaches for other large scale computer vision[7] and natural language processing[5] tasks. Future solutions will need to accurately detect text, perform optical character recognition (OCR) to recognize the text, and have the ability to parse the extracted text into a semi-structured

schema for additional downstream tasks such as classification, named entity recognition, and entity linking / normalization. The ideal software solution will need to exhibit strong few-shot performance and/or enable easy and effective fine-tuning for target domains. This paper focuses on methods to improve *robustness* for the foundational text recognition task, but touches on ideas that the author believes will be necessary to build robust end-to-end document processing solutions that can **fully automate** a variety of business workflows and remove tedious manual processing. The following sections will give a brief overview of the state of deep learning with a focus on applications in text recognition and then transition to synthetic data before continuing to the main contribution of this paper: data generating programs.

Chapter 3

Deep Learning and Text Recognition

3.1 Deep Learning

The success of deep learning across almost all major benchmarks in computer vision, natural language processing, speech recognition, and reinforcement learning tasks continues to reinforce Sutton's bitter lesson[35]: simple methods that leverage large amounts of data and computing power tend to outperform more complex and domain specific algorithms over a long enough time horizon. Research teams at Microsoft, Google, Facebook, and most recently OpenAI have shown that despite some theoretical limitations of deep learning based approaches, with excellent engineering work, deep learning continues to scale to new heights of performance by simply adding additional data and computing power to existing algorithms. In 2020, this is perhaps best exemplified by OpenAI's massive GPT-3 model that achieved state of the art performance on zero-shot and few-shot learning tasks within the natural language processing domain[5]. Research teams and individual researchers continue to innovate and show that there is still room for algorithmic innovation related to neural architecture design[29], activation functions[30], optimization algorithms[28], etc, but in general machine learning tasks are less constrained by algorithmic innovation than they were in the past, and this certainly seems to be true for text recognition models.

Machine learning practitioners in industry can often choose from an existing state of the art neural architecture (often with pre-trained models) and focus their efforts on data collection, annotation, and fine-tuning models through transfer learning. The author has found this to be true for text recognition, having leveraged both pretrained convolutional neural networks (CNNs) pretrained on ImageNet[33] and text recognition encoders / decoders pretrained[2] on MJSynth[21] and SynthText datasets[15].

Partially due to the open nature of machine learning research, the model architectures and training algorithms have become somewhat commoditized. Deep learning algorithms are able to automatically learn useful representations of input data and can then use these representations to make predictions, enabling transfer learning opportunities that did not exist in the past. The author speculates that the combination of proprietary data used to fine-tune models and the code to orchestrate the overall training, experimentation, and serving infrastructure may yield more competitive advantage than algorithmic innovation as of right now. There is some evidence for this, including the success of tools like AutoML[9], the investment in active learning and data management frameworks *without corresponding publications* (especially in autonomous vehicle research)[31], the emergence of data slicing and weak supervision based frameworks like Snorkel[32], and the rise of synthetic data and simulators in machine learning tasks[36]. It can be argued that many of the more recent advancements in machine learning have been due to our collective ability to harness greater and greater amounts of computing power paired with the collection of massive amounts of training data. This seems to be one of the fundamental barriers in the way of improving deep learning based tools – access to large amounts of data within a given domain. In many ways, collecting data has become the new barrier to progress in advancing machine learning technology, but this does not seem to be particularly true for text recognition models.

3.2 Deep Learning for Text Recognition

Text recognition, also known as optical character recognition, is the process of converting an image of text to a machine readable string of text and is a foundational piece of any automated document processing workflows based on computer vision techniques. This task was one of the first computer vision problems that artificial neural networks (ANNs) achieved success in[25] and is well suited for deep learning algorithms due to their ability to meaningful representations of visual features without requiring manual feature engineering.[4] A simplified version of this task (digit recognition) is one of the most common benchmarks in computer vision, and almost all machine learning practitioners have interacted with the MNIST dataset[25]. There have been many different neural architectures over the years, but researchers at Clova AI recently presented a unified framework for text recognition that capture the major components of most deep learning based text recognition models[2]

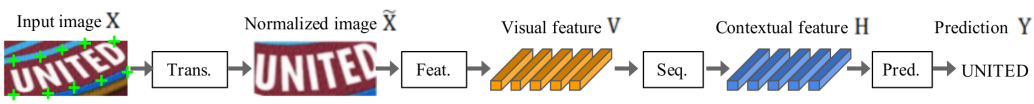
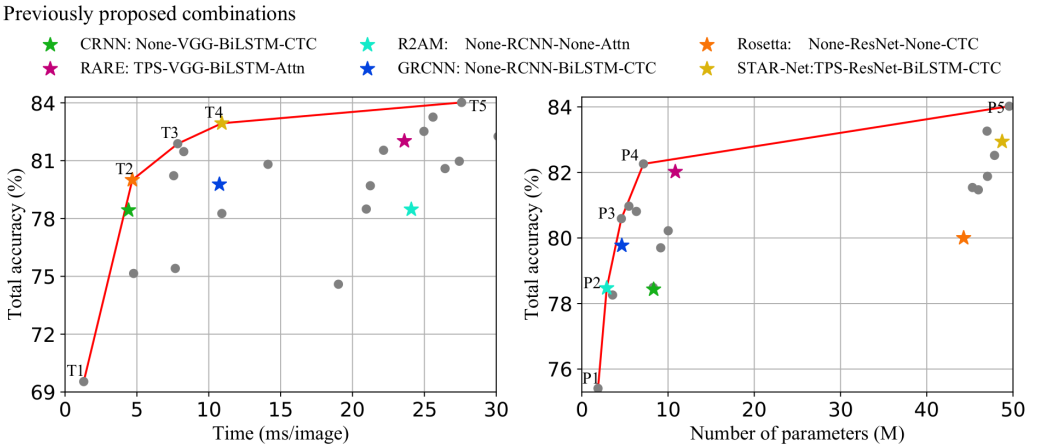


Figure 3: Visualization of an example flow of scene text recognition. We decompose a model into four stages.

In this framework, there are four distinct stages used to translate an image into

machine readable text. First, the image is optionally transformed to a unified spatial representation to simplify downstream stages[22]. This is mostly used for scene text recognition, and does not necessarily apply to more traditional document based OCR due to less variability in text shapes. Second, features are extracted from the text image, typically with some variant of a convolutional neural network (CNN) such as ResNet[16]. These visual features are then encoded into a compressed representation via a sequence modeling stage, typically a variant of a recurrent neural network (RNN) or a transformer[17][37]. Finally, the decoder stage makes predictions using either the visual features, encoded representations from the sequence model, or both. The final prediction stage typically either uses connectionist temporal classification (CTC)[14], attention based decoding[8], or a combination of the two as in more recent approaches[19]. Baek et al. provide a rigorous analysis of performance and accuracy tradeoffs, open-source code for training text recognition models, and a collection of pre-trained models with open access for other researchers. The choice of visual feature extraction architecture mostly impacts the overall memory footprint of the model due to the large number of parameters in modern ResNets, while the decoder / prediction mechanism mostly impacts inference speed and overall accuracy.



#	Trans.	Feat.	Seq.	Pred.	Acc. %	Time ms	params $\times 10^6$
T1	None	VGG	None	CTC	69.5	1.3	5.6
T2	None	**ResNet**	None	CTC	80.0	4.7	46.0
T3	None	ResNet	**BiLSTM**	CTC	81.9	7.8	48.7
T4	**TPS**	ResNet	BiLSTM	CTC	82.9	10.9	49.6
T5	TPS	ResNet	BiLSTM	**Attn**	84.0	27.6	49.6

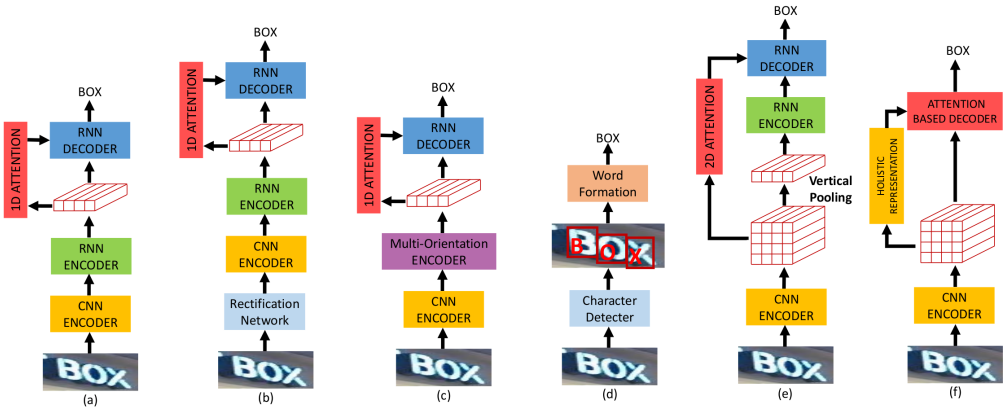
(a) Accuracy versus time trade-off curve and its frontier combinations

#	Trans.	Feat.	Seq.	Pred.	Acc. %	Time ms	params $\times 10^6$
P1	None	RCNN	None	CTC	75.4	7.7	1.9
P2	None	RCNN	None	**Attn**	78.5	24.1	2.9
P3	**TPS**	RCNN	None	Attn	80.6	26.4	4.6
P4	TPS	RCNN	**BiLSTM**	Attn	82.3	30.1	7.2
P5	TPS	**ResNet**	BiLSTM	Attn	84.0	27.6	49.6

(b) Accuracy versus memory trade-off curve and its frontier combinations

Yang et al. presented an alternative view of text recognition models in their paper "A Holistic Representation Guided Attention Network"[39], which the author finds particularly suitable for this task due to their usage of transformers and purely attention based decoding – enabling massive speedup in training due to parallelization capabilities. There are many factors that go into choosing the appropriate architecture for any particular use case, and the framework for choosing that architecture is outside the scope of this paper. The author has found success with a simple ResNet encoder coupled with an attention based decoder that attends over all visual features

and sequentially decodes from them. This architecture is resilient to variance in text detection accuracy and achieves strong overall accuracy, but this is at the cost of slower inference times. The author leaves architectural modifications that might speed up inference, such as guided CTC, for future research.



It seems that most of the innovation related to text recognition in recent years has focused on neural architectures, with recent work focusing on adapting transformer networks to the sequence modeling and decoding tasks[13]. Relatively little work has been published focusing on the underlying data used to train these models, despite the success of other researchers in scaling computer vision and natural language processing tasks with carefully curated and expansive data sets. Text recognition models are often trained on synthetic data, due to the high cost of manually annotating document images with bounding boxes and corresponding text strings. Synthetic data is particularly well suited to text recognition in the case of models designed to extract text from digital documents and scanned images of printed text due to the overlap in domain distribution. The author of this paper believes that intelligently expanding and improving training data combined with basic curriculum learning will yield performance improvements somewhat independently of architectural choices.

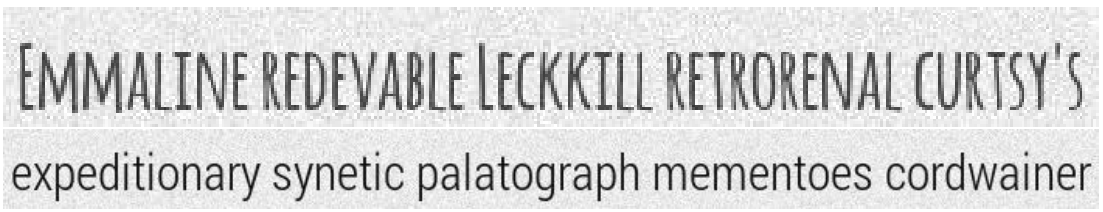
Chapter 4

Synthetic Data

4.1 Synthetic Data for Text Recognition

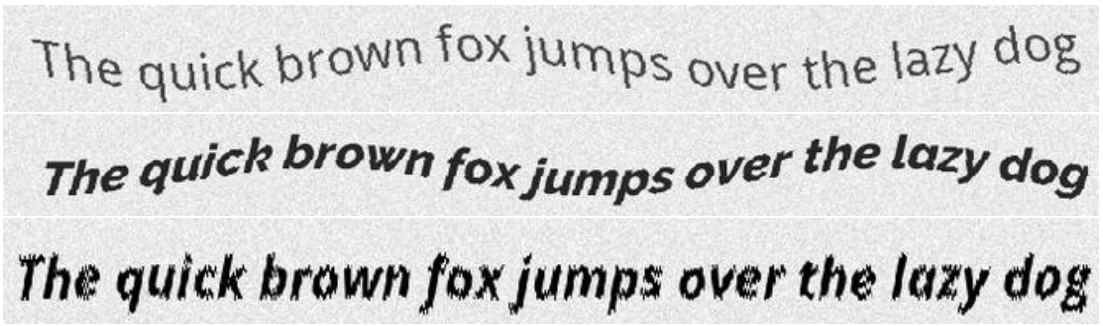
There are a few commonly used synthetic text image datasets available, but often machine learning practicioners will create pipeline code to generate their own dataset and attempt to align semantic and syntactic content with their target domain. The author has observed several dimensions of variability to consider when generating synthetic text images. The first choice to make is the actual text content that should be rendered. This could be pulled from a dictionary, generated via a programmatic approach such as a regular expression based generator, random characters, manually curated content, etc. Once the text content has been selected or generated, the font, font size, font color, (optionally) kernel distance between characters, and any other facets related to the shape of the characters must be chosen. Next, the text should be superimposed onto a background image which might consist of a single color, a gradient, a randomly chosen image, etc. Finally, transformations such as distortion, skewing, blurring, etc. can be applied to adjust the overall image. A few examples of this process can be seen below, sourced from the open-source Text Recognition Data Generator repository[3]

4.1.1 Fonts

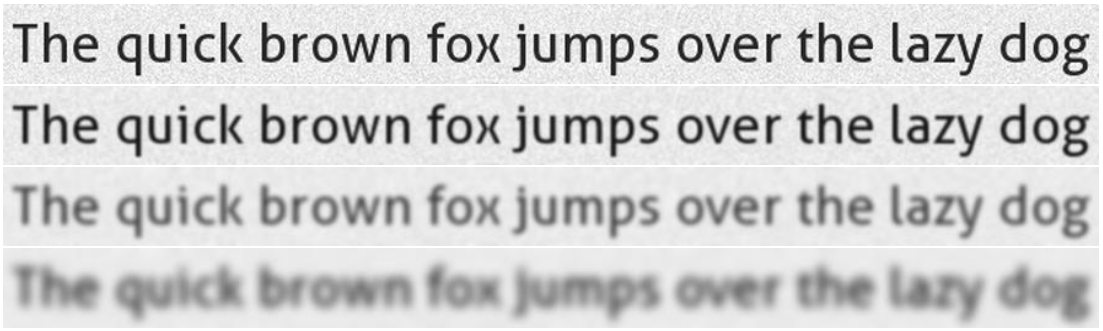


4.1.2 Text Skewing

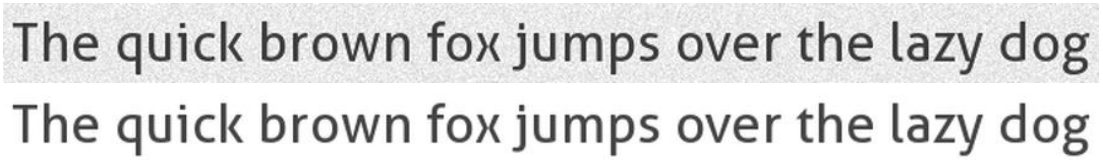
4.1.3 Text Distortion



4.1.4 Text Blurring



4.1.5 Background



4.2 Synthetic Data Literature Review

Synthetic data has emerged as a promising potential solution to the rising costs of data curation, with many successful applications in computer vision, robotics, and reinforcement learning tasks. Historically, there have been major challenges aligning synthetic data to real world data, limiting its potential impact. The last few years have seen several breakthroughs that have enabled synthetic data to be used more successfully in both academia and industry. Domain randomization in physics simulators enabled successful transfer learning from synthetic data to real world tasks and set a new state of the art result in object localization at the time of publishing[36]. Generative teaching networks enabled significantly faster training and neural architecture search by building a synthetic data curriculum for learners, but are restricted to a single meta-learned generator program[34]. Meta-Sim proposed a method to generate synthetic data that aligns synthetic data to a target domain content distribution resulting in task specific and more realistic synthetic data[23]. The author observes a common theme across these three separate innovations. First, the synthetic data need not be perfectly aligned to real-world data in order to produce useful models. Indeed, it seems that having some misalignment and randomization can produce more robust models, as in the case of domain randomization. This is reinforced outside the domain of synthetic data by the success of approaches such as data augmentation and adversarial example mining during training, and shows that increasing levels of data diversity can contribute to stronger performance[20]. Second, models need not be trained on realistic looking data (at least initially) as in the case of generative teaching networks. Finally, it seems that synthetic data must align on a content perspective to produce optimal results, even if there is not total alignment semantically. Keeping these ideas in mind, the author explores how to integrate aspects of the previously mentioned research towards text recognition.

Chapter 5

Evolutionary Computation

5.1 Genetic Programming

5.1.1 Overview

Genetic programming (GP) is a type of evolutionary algorithm used to synthesize one or more computer programs designed to solve a task[24]. GP works by initializing a population of (typically random) programs and then iterating through an evolutionary process to explore the search space of possible programs. A genetic programming algorithm usually includes initializing a population, evaluating the fitness of each program within the population, selecting individual programs based on their relative fitness as per the Darwinian principle of natural selection, and then applying genetic operators such as random mutation and crossover to evolve new programs. Genetic programming is an excellent algorithm for the use case detailed in this case study because it provides a computationally efficient mechanism to explore the massive search space of data generating programs for text recognition. In addition to computational efficiency, genetic programming also provides a strong mechanism to induce gradual complexity via genetic operators (as per the building blocks hypothesis) which has the potential to induce novelty and diversity in the population[18]. The final result of a GP run should be one or more individual programs that can be embedded in other applications.

5.1.2 Individual Programs

The individual programs created by genetic programming are comprised of terminals and primitive functions. Terminals include input variables, random constants, and functions that do not accept arguments. Primitive functions are low level functions

that can be applied to terminals – for example, mathematical operators such as subtraction, addition, multiplication, and division. There are several different structures that can be used for genetic programming such as linear methods, Cartesian graphs, stack-based methods, etc. but tree based structures are the most common. In this form of GP, synthesized programs are represented as a tree which is executed recursively until all nodes have been evaluated. Terminals and primitive functions are composed together to produce a program which can be executed to produce some output. There are a variety of mechanisms that can be used to synthesize these programs, but genetic programming will always include an initial program synthesis phase which creates a population of programs and an evolutionary process phase that iterates through the program search space.

5.1.3 Evolution

Once an initial population of programs has been created, each program is individually evaluated using some pre-defined criteria that produces a fitness score. For example, the fitness score might measure the accuracy of predictions made by the program on some task. After the population of programs have all been evaluated based on their fitness score, the individual programs are probabilistically selected for advancement to the next stage relative to their fitness score. Individual programs with higher fitness scores are more likely to be selected for advancement, while programs with lower comparative fitness scores are less likely to advance. Once a program has been selected, some genetic operator is applied to the program to modify it. This could be a random mutation of one or more of the terminals or primitive functions, or it could be a more complex operator that breeds programs together through genetic crossover by mixing and matching elements from each program. A full overview of the evolutionary process is outside the scope of this paper, as only random mutation was applied as a genetic operator. This process repeats until a stopping condition is met, such as the number of steps in a run, and the process concludes with the final list of programs.

5.2 Novelty Search

The fitness function used in genetic programming is typically a heuristic used to measure progression towards a task specific objective and is the basis for optimization. The assumption is that by optimizing directly on the fitness function this will create evolutionary selection pressure yielding increasing complexity in programs which consequently yield higher and higher performance on the target task. There is some debate in the evolutionary computation community around this, but one credible hypothesis is that optimizing on the fitness score is not the mechanism that produces additional complexity, but that genetic drift naturally produces this[26]. It is possible

that the pressure to maximize the fitness score of an individual can encourage convergence to a local minimum in the case where the fitness landscape is deceptive. Novelty search provides us with a potential solution to ameliorate the problem of deceptive objectives. It is an excellent fitness function when considering the goal of creating a diverse population of data generating programs. Novelty search explicitly optimizes for novel behaviour (in some measurable dimension) rather than explicitly optimizing for the target tasks, and run over enough iterations should yield population level diversity as a consequence. The core concept in novelty search is to evolutionarily privilege divergent behaviour compared with previous programs rather than explicitly rewarding performance relative to an objective tailored towards the end goal.

5.3 Minimal Criterion

Minimal criterion acts as a modification on top of novelty search to reduce the search space of behaviours to viable programs and improve computational efficiency during the evolutionary process. This modification requires that any synthesized program must meet some minimal criterion in order to be allowed to reproduce – any program not meeting the minimal criterion is assigned a fitness score of 0 and removed from the evolutionary pool for breeding[27]. This method is particularly effective when combined with novelty search because it requires programs to meet the minimal criteria required to survive **in a novel way**. Searching for novel ways to meet the same minimal criteria can result in a more diverse set of foundational building blocks that can be branched off from, resulting in greater genetic variety and potentially more effective genetic drift.

Chapter 6

Data Generating Programs

In this section, the author proposes that increasing data diversity with respect to learned representations will result in more robust models, outlines the algorithm for data generating programs that produce synthetic data for text recognition, and details some of the modifications required to genetic programming in order to successfully implement the overall algorithm.

6.1 Diverse Data Hypothesis

Text recognition is a particularly interesting task to consider given the recent developments in deep learning at scale and their reliance on artificially generated training data. In many ways, the ability to generate synthetic data removes the limitations / cost related to curating labeled training data. Machine learning practitioners have the ability to generate a nearly infinite set of training data and can expect a reasonable overlap between synthetic data and real world documents. This shifts the constraint from data curation to compute cost, and gives rise to the following challenge. Within the context of a machine learning task, given the ability to generate a practically infinite amount of synthetic data but a finite amount of computing power and time resources, how to build a maximally robust model for the task? Inspired by the success of ensemble based approaches that promote *model diversity* such as decision tree ensembles[6], the author proposes focusing on *data diversity* as articulated below.

Diverse Data Hypothesis: Encouraging high levels of diversity across a meaningful dimension during synthetic data generation for text recognition can yield robust models that are able to adapt to new data distributions and permutation variations

The practical application of this idea for the domain of text recognition is to generate lots of different and interesting text data generating programs, use them to generate synthetic training data, and then train text recognition models on the diverse training data. This requires mechanisms to measure both the overall diversity of the data generating programs population and the novelty of an individual data generating program. Text data generating programs have a massive search space that would be infeasible to explore, and a tremendous amount of semantic overlap between different configurations. For example, choosing two similar fonts would produce roughly the same images and fail to produce useful diversity despite potentially completely different genotypic encodings. The author leveraged genetic programming, an evolutionary computation framework, to provide program synthesis and to intelligently explore the search space of synthetic data generating programs for text recognition models, with an adapted fitness function designed to encourage gradual emergent complexity and to optimize indirectly for population diversity.

6.2 Data Generating Programs for Text Recognition

Data Generating Programs are an algorithmic approach designed to integrate components of the existing research reviewed and to test the proposed diverse data hypothesis in a practical context with the goal of producing more robust text recognition models for document vision tasks. The DGP algorithm leverages a customized variant of genetic programming to provide program synthesis, novelty search to encourage exploration of a semantically useful search space, and applies a minimal criterion to ground the resulting output to be realistic enough. First, a pre-trained visual feature extractor is initialized (in this case a ResNet-34 model pre-trained on ImageNet). Second, a pre-trained text recognition model is initialized. Then, a population of randomly generated synthetic data generating programs is initialized. That population goes through an evolutionary process based on novelty search as articulated previously. Finally, the final population of data generating programs is sampled from to create a synthetic text image training dataset and the text recognition model is fine-tuned on this data.

6.3 Algorithm for DGP

Algorithm 1: Data Generating Programs for Text Recognition

Result: Text Recognition Model and Archive of Data Generating Programs

Initialization:;

1. Load pre-trained visual feature extractor (i.e. resnet-34);
2. Load pre-trained text recognition model (i.e. rosetta);
3. Instantiate archive A to index synthesized data generating programs;
4. Instantiate population P of data generating programs, having size N;

while *counter* < *generations* **do**

1. **for** $i \leftarrow 0$ **to** N 1 **do**
   - Extract behavioural characterization from DGP using visual feature extractor;
   - Test if DGP meets minimal criteria using text recognition model;
   - Calculate fitness score based on novelty;
   - Add DGP to archive if minimal criteria is met;

end

2. Select individuals with a bias towards fitness score;
3. Apply genetic operators to individuals;

end

Construct final population of DGPs via KNN algorithm;

Generate synthetic training data by uniformly sampling from from final population;

Train text recognition model on synthetic data for L iterations;

6.4 Genetic Programming for Text Recognition Data Generators

A data generating program (DGP) for synthetic text images requires a program with several components and some type constraints in order to produce usable output. The author approaches this problem by breaking program synthesis from a single program into an aggregation of multiple sub-modules. First, the DGP must create a background to superimpose text visuals onto. Ideally this module should encourage a variety of different backgrounds to yield stronger visual diversity. Second, the DGP must either accept text content as an input or generate it's own text content. Third, the DGP must print the actual text content visually choosing a font, color, orientation, etc. Finally, the DGP can optionally compose together a series of transformations to alter the overall image (i.e. blurring, color inversions, rotations, etc.)

Traditionally, genetic programming focuses on finding a single best individual for the given task that is being optimized for in a given run. Leveraging this type of algorithm to build a diverse population of text data generating programs requires some

adaptations to the overall algorithm. First, the algorithm must be adapted to deal with sub-modules and constrain genetic operators accordingly to ensure they are properly applied. Second, the algorithm must enforce type constraints and an overall structure to yield valid text data generating programs. For example, there must always be a background (even if it's only a single color), and there must always be a way to select / generate text content. Third, the algorithm should be encouraged to produce useful programs and avoid spending computational power on useless programs. For example, if a DGP blurs an image so much that none of the text content could possibly be recognized then it should be discarded. Finally, we need to define a fitness function to optimize for that will yield novel types of programs in order to build a diverse population of synthetic data generators. Novelty search provides us with a fitness function mechanism to optimize for diversity indirectly, and applying a minimal criterion to generated programs encourages computational efficiency by pruning the active search space that is explored.

6.5 Novelty Search for DGP

The author considered using a standard fitness function that measured the quality of the data generated relative to the performance of the text recognition model. This idea was discounted due to the computational burden this would add (constantly training models on every data generating programs outputs), and an intuition that the reward space is deceptive around synthetic data generators and optimal levels of diversity would not be achieved thus potentially failing to achieve robustness. In this algorithm, a mechanism to measure the behaviour of the individual programs is required along with a method to calculate novelty with respect to the past and present population of data generating programs. In order to measure the behaviour of data generating programs, the author selected a set of text contents that align with their target domain tasks (i.e. email addresses, dates, invoice numbers, etc.) and extracts visual features using a pre-trained ResNet (first stage of text recognition model). These visual features are the learned representations that the encoder / decoder stages attend to, and provide a meaningful way to measure novelty of an individual program. In this way, the author avoids focusing on genotypic variation and shrinks the search space to focus on more useful variation (i.e. Arial vs. Arial Regular fonts are different genotypically, but produce roughly the same visual output). Finally, the author used a variant of the k-nearest neighbors algorithm to measure novelty of data generating programs as per the original novelty search algorithm[26]

$$\rho(x) = \frac{1}{k} \sum_{i=0}^{k} \text{dist}\,(x, \mu_i)$$

This function measures the sparseness of a particular point in the behavioural characterization space and rewards programs that manage to reach previously unexplored

spaces. The measure of sparseness is the average distance to the k-nearest neighbors, where distance is measured using the cosine similarity of the two vectors. In this way, the algorithm is incentivized to synthesize new programs that explore new spaces in the search space and diverge from past behaviour.

6.6 Minimal Criterion for DGP

The idea of applying a minimal criterion to data generating programs provides a simple but effective constraint on the algorithm to produce data generating programs which can produce data that is realistic enough so as to be useful. This is a simple modification to the novelty search algorithm, but extremely useful for the use case of synthesizing data generating programs for text recognition given the use case laid out by the author. If this constraint was not applied, it is very likely that increasingly diverse data generating programs would be synthesized, but that they would not actually be useful for generating training data for the goal. The author leverages the following minimal criterion during the evolutionary process: A pre-trained text recognizer model must be able to recognize 10 percent or more of the characters in the synthetic image correctly.

Chapter 7

Experiments

7.1 Datasets

The datasets, code, and artifacts produced in the course of this research are privately curated and closed source, but the author is able to share high level results / metrics related to experiment runs. The author curated a collection of invoice and resume documents, cropped each word into new images based on detected bounding boxes from an open-source text detection model, and annotated the text images with the appropriate text content label. The author then further split the text images from invoice documents into a training dataset and a testing dataset to allow for optional fine-tuning on the invoice domain and evaluate performance across multiple document categories. The author sourced approximately twice as many invoice documents in order to test transfer performance to a new domain, but the author also evaluates performance on a subset of invoice documents to compare performance across models fine-tuned on the target domain vs. models trained purely on synthetic data. Finally, the author provides an evaluation on a combination of both invoices and resumes in aggregate. In this manner, the author was able to test robustness of the tech recognition model – or how well it transferred performance to a new domain trained purely on synthetic data. The author tested four major models using a standard neural architecture, detailed below. This resulted in a dataset of four thousand word images total, two thousand coming from resume images and two thousand coming from invoice images. As mentioned previously, there was an additional set of two thousand images coming from invoice images available for fine-tuning the baseline model.

7.2 Neural Architecture

The author leverages an open source implementation of the following architecture taken from "A Holistic Representation Guided Attention Network"[39]. The first stage of the neural architecture is a modified convolutional neural network (CNN) based on the ResNet34 architecture. The visual representation extractor encodes the text image into both a 2D feature map and a 1D holistic representation. The holistic representation forms the basis for behavioural characterization, while the model primarily learns to attend across the 2D feature map while being guided by the holistic representation. All other details are identical to the original implementation.

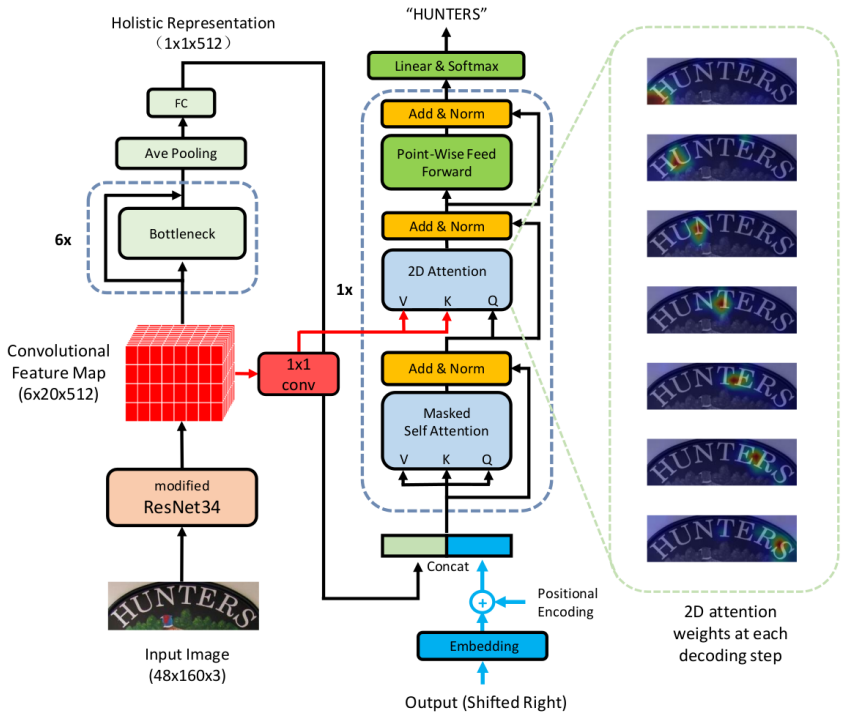


Figure 2: The overall structure of our proposed model. It consists of two parts: a ResNet34-based image encoder (left) and an attention-based sequence decoder (right). The 2D feature maps generated by modified ResNet34 are connected to the 2D attention module in the decoder by a 1×1 convolution layer, and we stack 6 bottleneck modules to extract the 1D holistic representation which can help the 2D attention more accurate. The bottleneck module is the same as in ResNet [36]. In contrast to other irregular text recognizers [37, 7], there is no recurrent networks to model the representation. As a non-recurrent network, our model can be trained in parallel. Furthermore, training our model only needs word-level annotations.

7.3 Metrics

In order to evaluate quality of text recognition models, the author uses overall accuracy on words, only counting a word as correct if it is exactly correct. This departs from

traditional methods of evaluating text recognition models – most research focuses on distance edits using an algorithm like the levenshtein distance formula, but for real world business workflows, complete accuracy is required, especially on key fields such as email addresses, invoice numbers, amounts due, etc.

7.4 Text Recognition Models

The author ran the following experiments to test the diverse data hypothesis and attempt to improve baseline text recognition models.

- **Model #1:** Baseline (Pre-Trained Text Recognizer)
- **Model #2:** Fine-Tuned on Invoice Training Dataset
- **Model #3:** Additional Training on Random Synthetic Data
- **Model #4:** Additional Training on Data from DGP Algorithm

Only model 3 and model 4 were trained on synthetic data generated as part of this project. To create the training dataset for model 3, one hundred data generating programs were randomly initialized and twenty thousand total images were sampled from them. To create the training dataset for model 4, one hundred data generating programs were evolved using the data generating programs algorithm outlined previously. The GP process was run for five hundred iterations and the resulting one hundred programs were sampled from to create twenty thousand training images. All four models are evaluated on the following datasets:

- **Invoices:** Text Images from Invoice Documents (Test Dataset)
- **Resumes:** Text Images from Resume Documents
- **Combined:** Class Balanced Combination of Text Images from both Invoice and Resume Documents

7.5 Results

Each model is evaluated on accuracy on the test data set, as defined by the overall number of words it recognizes correctly divided by the total number of words to be predicted, only counting a word as correct if it is exactly correct. This metric was chosen to reflect the binary usefulness of a text recognition model for information extraction tasks – a model must be exactly correct in order to avoid manual intervention. There are four thousand images in total, two thousand for images of words coming from invoice documents and two thousand for images of words coming from resume documents.

Table 7.1: Experiment Results - Accuracy

Model	Invoice Images	Resume Images	Combined Images
Model #1: Baseline	62.35%	55.05%	58.70%
Model #2: Fine-Tuned on Invoices	85.10%	43.25%	64.18%
Model #3: Random Synthetic Data	48.15%	28.30%	33.23%
Model #4: Data Generating Programs	79.15%	63.40%	71.28%

As might be expected, the model which is fine-tuned on images coming from invoice documents outperforms all other models on invoice images, but it appears to lose some robustness to other domains during the fine-tuning process. The model which is trained on randomly generated synthetic data under-performs the baseline, and is significantly worse on images coming from resume documents, likely due to the greater difficulty of these types of images. The model which is trained on data on data produced by the data generating program algorithm produces the most robust model in terms of accuracy across domains, but does not meet the performance of the model fine tuned on invoice images.

Chapter 8

Conclusion

The author presents a novel approach to synthetic data generation for text recognition models, mainly building a diverse population of data generating programs using genetic programming with novelty search, and sampling from the program population to create training data. This project was designed as a case study to examine the impact of focusing on training data rather than modifying the learning algorithm directly. This approach yielded a more robust text recognition model, at the cost of losing some in-domain performance. The final model produced based on the articulated approach has been integrated into a document vision pipeline that is used in a production environment. The author believes that while more experimentation is called for, there appears to be some usefulness in the idea of diverse data generators, and believes that novelty search combined with evolutionary computation methods is an excellent way to build these populations. This idea is not restricted to genetic programming, the only prerequisite is that the data generating programs can gradually accumulate complexity and can produce realistic enough data as to be useful. The datasets, code, and artifacts produced in the course of this research are privately curated and closed source, and the author is unfortunately not permitted to share them publicly. The author looks forward to creating public datasets, a more robust open source implementation, and sharing both with the broader machine learning community in the future. The author also intends to re-implement and open source document vision models to improve the open-source text recognition models available to other researchers.

The initial results produced are very encouraging, despite some limitations / challenges. The author believes that the following areas would be interesting to explore as modifications to the core algorithm or extensions on top of it. First, the set of primitives used for program synthesis are somewhat limiting. Adding in more diverse background generators and text content generators would open the doors to stronger

diversity and increasing complexity. Second, this approach assumes a single global model that is trained on a diverse set of data. It might yield better performance to train an ensemble of models on datasets produced by one or more data generating programs and combine these models through a generative or knowledge distillation based process. Third, it seems clear that there is an opportunity to leverage fine-tuning on the target domain based on the loss of performance on in-domain tasks. It is possible that some degree of catastrophic forgetting is occurring – a few possible remediations for this would be to explore a neural network with modulated neuroplasticity, add a regularization method that encourages sparsity, or some other mechanism to reduce catastrophic forgetting. Fourth, there has been some very interesting development in contrastive learning approaches, particularly for computer vision tasks. An alternative approach to consider would be to add a co-evolutionary mechanism that encourages permutation invariant representations across the same text content – perhaps focusing on character level alignment across different data generating programs could yield interesting results.

Finally, the author believes that there is some intersection of data set slicing / active learning, co-evolutionary mechanisms, and meta-learning that could be combined in a more holistic process. The author imagines an algorithm that can train on a diverse set of data to build a strong baseline and then provide a mechanism for domain adaptation that leverages evolving new data generating programs and modulating the learning environment to optimally adapt to new domains while retaining knowledge of previously seen domains. The author looks forward to exploring some of these ideas in the future.

Bibliography

[1] *Amazon Textract | Extract Text & Data | AWS*. en-US. URL: https://aws.amazon.com/textract/ (visited on 08/31/2020).

[2] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee. "What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis." In: *arXiv:1904.01906 [cs]* (Dec. 2019). arXiv: 1904.01906 version: 4. URL: http://arxiv.org/abs/1904.01906 (visited on 08/31/2020).

[3] E. Belval. *Belval/TextRecognitionDataGenerator*. original-date: 2017-07-01T20:11:40Z. Aug. 2020. URL: https://github.com/Belval/TextRecognitionDataGenerator (visited on 08/31/2020).

[4] Y. Bengio, A. Courville, and P. Vincent. "Representation Learning: A Review and New Perspectives." In: *arXiv:1206.5538 [cs]* (Apr. 2014). arXiv: 1206.5538. URL: http://arxiv.org/abs/1206.5538 (visited on 08/31/2020).

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. "Language Models are Few-Shot Learners." In: *arXiv:2005.14165 [cs]* (July 2020). arXiv: 2005.14165. URL: http://arxiv.org/abs/2005.14165 (visited on 08/31/2020).

[6] H. Chen, H. Zhang, S. Si, Y. Li, D. Boning, and C.-J. Hsieh. "Robustness Verification of Tree-based Models." In: *arXiv:1906.03849 [cs, stat]* (Dec. 2019). arXiv: 1906.03849. URL: http://arxiv.org/abs/1906.03849 (visited on 08/31/2020).

[7] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton. "Big Self-Supervised Models are Strong Semi-Supervised Learners." In: *arXiv:2006.10029 [cs, stat]* (June 2020). arXiv: 2006.10029. URL: http://arxiv.org/abs/2006.10029 (visited on 08/31/2020).

[8] Z. Cheng, F. Bai, Y. Xu, G. Zheng, S. Pu, and S. Zhou. "Focusing Attention: Towards Accurate Text Recognition in Natural Images." In: *2017 IEEE International Conference on Computer Vision (ICCV)* (Oct. 2017). arXiv: 1709.02054,

pp. 5086–5094. DOI: 10.1109/ICCV.2017.543. URL: http://arxiv.org/abs/1709.02054 (visited on 08/31/2020).

[9] *Cloud AutoML - Custom Machine Learning Models*. en. URL: https://cloud.google.com/automl (visited on 08/31/2020).

[10] *Computer Vision | Microsoft Azure*. en. URL: https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/ (visited on 08/31/2020).

[11] C. Cutter. "Résumés Are Starting to Look Like Instagram—and Sometimes Even Tinder." en-US. In: *Wall Street Journal* (Aug. 2019). ISSN: 0099-9660. URL: https://www.wsj.com/articles/resumes-are-starting-to-look-like-instagramand-sometimes-even-tinder-11565707364 (visited on 08/31/2020).

[12] *Document AI*. en. URL: https://cloud.google.com/solutions/document-ai (visited on 08/31/2020).

[13] X. Feng, H. Yao, Y. Qi, J. Zhang, and S. Zhang. "Scene Text Recognition via Transformer." In: *arXiv:2003.08077 [cs]* (Apr. 2020). arXiv: 2003.08077. URL: http://arxiv.org/abs/2003.08077 (visited on 08/31/2020).

[14] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks." In: *Proceedings of the 23rd international conference on Machine learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, June 2006, pp. 369–376. ISBN: 9781595933836. DOI: 10.1145/1143844.1143891. URL: https://doi.org/10.1145/1143844.1143891 (visited on 08/31/2020).

[15] A. Gupta, A. Vedaldi, and A. Zisserman. "Synthetic Data for Text Localisation in Natural Images." In: *arXiv:1604.06646 [cs]* (Apr. 2016). arXiv: 1604.06646. URL: http://arxiv.org/abs/1604.06646 (visited on 08/31/2020).

[16] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition." In: *arXiv:1512.03385 [cs]* (Dec. 2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385 (visited on 08/31/2020).

[17] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory." In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (visited on 08/31/2020).

[18] J. Holland and H. Mimnaugh. *Hidden Order: How Adaptation Builds Complexity*. Basic Books, Sept. 1996.

[19] W. Hu, X. Cai, J. Hou, S. Yi, and Z. Lin. "GTC: Guided Training of CTC Towards Efficient and Accurate Scene Text Recognition." In: *arXiv:2002.01276 [cs, eess]* (Feb. 2020). arXiv: 2002.01276 version: 1. URL: http://arxiv.org/abs/2002.01276 (visited on 08/31/2020).

[20] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. "Adversarial Examples Are Not Bugs, They Are Features." In: *arXiv:1905.02175 [cs, stat]* (Aug. 2019). arXiv: 1905.02175. URL: http://arxiv.org/abs/1905.02175 (visited on 08/31/2020).

[21] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. "Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition." In: *arXiv:1406.2227 [cs]* (Dec. 2014). arXiv: 1406.2227. URL: http://arxiv.org/abs/1406.2227 (visited on 08/31/2020).

[22] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. "Spatial Transformer Networks." In: *arXiv:1506.02025 [cs]* (Feb. 2016). arXiv: 1506.02025. URL: http://arxiv.org/abs/1506.02025 (visited on 08/31/2020).

[23] A. Kar, A. Prakash, M.-Y. Liu, E. Cameracci, J. Yuan, M. Rusiniak, D. Acuna, A. Torralba, and S. Fidler. "Meta-Sim: Learning to Generate Synthetic Datasets." In: *arXiv:1904.11621 [cs]* (Apr. 2019). arXiv: 1904.11621. URL: http://arxiv.org/abs/1904.11621 (visited on 08/31/2020).

[24] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992. ISBN: 0-262-11170-5. URL: http://mitpress.mit.edu/books/genetic-programming.

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324. ISSN: 1558-2256. DOI: 10.1109/5.726791.

[26] J. Lehman and K. O. Stanley. "Abandoning Objectives: Evolution Through the Search for Novelty Alone." In: *Evolutionary Computation* 19.2 (Sept. 2010), pp. 189–223. ISSN: 1063-6560. DOI: 10.1162/EVCO_a_00025. URL: https://doi.org/10.1162/EVCO_a_00025 (visited on 08/31/2020).

[27] J. Lehman and K. O. Stanley. "Revising the evolutionary computation abstraction: minimal criteria novelty search." In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. GECCO '10. Portland, Oregon, USA: Association for Computing Machinery, July 2010, pp. 103–110. ISBN: 9781450300728. DOI: 10.1145/1830483.1830503. URL: https://doi.org/10.1145/1830483.1830503 (visited on 08/31/2020).

[28] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han. "On the Variance of the Adaptive Learning Rate and Beyond." In: *arXiv:1908.03265 [cs, stat]* (Apr. 2020). arXiv: 1908.03265. URL: http://arxiv.org/abs/1908.03265 (visited on 08/31/2020).

[29] S. Merity. "Single Headed Attention RNN: Stop Thinking With Your Head." In: *arXiv:1911.11423 [cs]* (Nov. 2019). arXiv: 1911.11423. URL: http://arxiv.org/abs/1911.11423 (visited on 08/31/2020).

[30] D. Misra. "Mish: A Self Regularized Non-Monotonic Activation Function." In: *arXiv:1908.08681 [cs, stat]* (Aug. 2020). arXiv: 1908.08681. URL: http://arxiv.org/abs/1908.08681 (visited on 08/31/2020).

[31] *PyTorch at Tesla - Andrej Karpathy, Tesla - YouTube*. URL: https://www.youtube.com/watch?v=oBklltKXtDE (visited on 08/31/2020).

[32] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. "Snorkel: Rapid Training Data Creation with Weak Supervision." In: *Proceedings of the VLDB Endowment* 11.3 (Nov. 2017). arXiv: 1711.10160, pp. 269–282. ISSN: 21508097. DOI: 10.14778/3157794.3157797. URL: http://arxiv.org/abs/1711.10160 (visited on 08/31/2020).

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge." In: *arXiv:1409.0575 [cs]* (Jan. 2015). arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575 (visited on 08/31/2020).

[34] F. P. Such, A. Rawal, J. Lehman, K. O. Stanley, and J. Clune. "Generative Teaching Networks: Accelerating Neural Architecture Search by Learning to Generate Synthetic Training Data." In: *arXiv:1912.07768 [cs, stat]* (Dec. 2019). arXiv: 1912.07768. URL: http://arxiv.org/abs/1912.07768 (visited on 08/31/2020).

[35] *The Bitter Lesson*. URL: http://incompleteideas.net/IncIdeas/BitterLesson.html (visited on 08/31/2020).

[36] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World." In: *arXiv:1703.06907 [cs]* (Mar. 2017). arXiv: 1703.06907. URL: http://arxiv.org/abs/1703.06907 (visited on 08/31/2020).

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention Is All You Need." In: *arXiv:1706.03762 [cs]* (Dec. 2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762 (visited on 08/31/2020).

[38] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou. "LayoutLM: Pre-training of Text and Layout for Document Image Understanding." In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Aug. 2020). arXiv: 1912.13318, pp. 1192–1200. DOI: 10.1145/3394486.3403172. URL: http://arxiv.org/abs/1912.13318 (visited on 08/31/2020).

[39] L. Yang, P. Wang, H. Li, Z. Li, and Y. Zhang. "A Holistic Representation Guided Attention Network for Scene Text Recognition." In: *arXiv:1904.01375 [cs]* (July 2020). arXiv: 1904.01375. URL: http://arxiv.org/abs/1904.01375 (visited on 08/31/2020).