TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)

Abstract
TensorFlow is an interface for expressing machine learning algorithms. It integrates
sequence prediction [47], move selection for Go [34], pedestrian detection [2], reinforcement
learning [38], and much more. TensorFlow features a flexible and user-friendly
programming model, and can run on a variety of hardware, from CPUs to GPUs to
customized devices. TensorFlow is open-source and can be explored and contributed
for by anyone, wherever they are located.

Authors:
Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro,
Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow,
Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser,
Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray,
Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar,
Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals,
Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng
Google Research

The Unreasonable Effectiveness of Recurrent Neural Networks

There’s something magical about Recurrent Neural Networks (RNNs). I still remember when I trained my first recurrent network for Image Captioning. Within a few dozen minutes of training my first baby model (with rather arbitrarily-chosen hyperparameters) started to generate very nice looking descriptions of images that were on the edge of making sense. Sometimes the ratio of how simple your model is to the quality of the results you get out of it blows past your expectations, and this was one of those times. What made this result so shocking at the time was that the common wisdom was that RNNs were supposed to be difficult to train (with more experience I’ve in fact reached the opposite conclusion). Fast forward about a year: I’m training RNNs all the time and I’ve witnessed their power and robustness many times, and yet their magical outputs still find ways of amusing me.

This post is about sharing some of that magic with you.

We’ll train RNNs to generate text character by character and ponder the question “how is that even possible?”

By the way, together with this post I am also releasing code on Github that allows you to train character-level language models based on multi-layer LSTMs. You give it a large chunk of text and it will learn to generate text like it one character at a time. You can also use it to reproduce my experiments below. But we’re getting ahead of ourselves; What are RNNs anyway?

Build the LSTM model

```python
# build the model: a single LSTM
print('Build model...')
model = Sequential()
model.add(LSTM(128, input_shape=(maxlen, len(chars))))
model.add(Dense(len(chars)))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')

print("Compiling model complete...")

Build model...
Compiling model complete...
```

Machine Learning

- Grew out of work in AI
- New capability for computers
Alice was excited!

Lots of tutorials
Loads of resources
Endless examples
Fast paced research
How to even data science?
How to even data science?

https://miro.medium.com/max/1552/1*Nv2NNALuokZEcV6bYEHdGA.png
Challenge

How to make this work in the real world?
Machine Learning’s Surprises

A Checklist for Developers when Building ML Systems
Hi, I’m Jade Abbott

My first tattoo will probably be:

tar -xvzf file.tar.gz

Then I’ll never need to look it up again

5:54 AM - 19 Mar 2019

1,847 Retweets 11,767 Likes
Surprises while...

Trying to deploy the model

After deployment of model

Trying to improve the model
Some context

❖ I won’t be talking about training machine learning models
❖ I won’t be talking about which models to chose
❖ I work primarily in deep learning & NLP
❖ I am a one person ML team working in a startup context
❖ I work in a normal world where data is scarce and we need to collect more
The Problem

I want to meet...

I can provide...

Embedding + LSTM + Downstream NN

Yes, they should meet

No they shouldn't
The Problem

I want to meet...
someone to look after my cat

I can provide...
- pet sitting
- cat breeding
- software development
- chef lessons

Language Model + Downstream Task

Yes, they should meet

No they shouldn’t
The Problem

I want to meet...

someone to look after my cat

I can provide...

pet sitting
cat breeding
software development
chef lessons

Yes, they should meet

No they shouldn't
The Problem

I want to meet...

someone to look after my cat

I can provide...

pet sitting
cat breeding
software development
chef lessons

The Model

Yes, they should meet

No they shouldn’t
Surprises trying to deploy the model
Expectations

- Train & evaluate model
- API
- CI/CD
- User testing
- Unit Tests
- Model

Train & evaluate model

API

CI/CD

User testing

Unit Tests

Model
Is the model good enough?
75% Accuracy
Performance Metrics

❖ Business needs to understand it
❖ Active discussion about pros & cons
❖ Get sign off
❖ Threshold selection strategy
Surprise #2

Can we trust it?
Skin Cancer Detection


Husky/Dog Classifier
Skin Cancer Detection


Husky/Dog Classifier
Explanations

Text with highlighted words:
I want to meet a photographer. I can provide photography.

Text with highlighted words:
I can provide snooker coaching. I want to meet an excel tutor with expertise in financial modeling.
“Why Should I Trust You?”
Explaining the Predictions of Any Classifier

Marco Tulio Ribeiro
University of Washington
Seattle, WA 98195, USA
marcotcr@cs.uw.edu

Sameer Singh
University of Washington
Seattle, WA 98195, USA
sameer@cs.uw.edu

Carlos Guestrin
University of Washington
Seattle, WA 98195, USA
guestrin@cs.uw.edu

ABSTRACT
Despite widespread adoption, machine learning models remain nearly black boxes. Understanding the reasons behind predictions is, however, quite important in many domains, including healthcare, finance, education, and advertising. Our work has focused on the problem of explaining the predictions of any classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction. We also propose a method to explain models by presenting representative individual predictions and their explanations in a non-redundant way, framing the task as a combinatorial optimization problem. We demonstrate the flexibility of these methods by explaining different models for text (e.g., random forests) and image classification (e.g., neural networks). We show the utility of explanations via novel experiments, both simulated and with human subjects, on various scenarios that require trust: deciding if one should trust a prediction, choosing between models, improving an uninterpretable classifier, and identifying why a classifier should not be trusted.

1. INTRODUCTION
Machine learning is at the core of many recent advances in science and technology. Unfortunately, the importance of how much the human understands a model’s behaviour, as opposed to seeing it as a black box.

Determining trust in individual predictions is an important problem when the model is used for decision making. When using machine learning for medical diagnosis (e.g., detecting cancer), it is important to assess the outputs on a case-by-case basis, as the consequences may be catastrophic.

Apart from trusting individual predictions, there is a need to evaluate the model as a whole before deploying it “in the wild”. To make this decision, users need to be confident that the model will perform well on real-world data, according to the metrics of interest. Currently, models are trained using accuracy metrics on an available validation dataset. However, real-world data is often significantly different, and further, the evaluation metric may not be indicative of the model’s goal. Importing individual predictions and their explanations in a workable solution will allow users to assess the model in question.

In this paper, we propose providing explanations for individual predictions to evaluate the model’s performance and select the models (and explainations) as solutions to the “trusting the model” problem. Our main contributions are summarized as follows:

* LIME, an algorithm that can explain the predictions of any classifier (or regressor) in a faithful way, by approximating it locally with an interpretable model.
* SP-LIME, a method that selects a set of representative cases.
Surprise #3

Will this model harm users?
“Racial bias in a medical algorithm favors white patients over sicker black patients”
“Racist robots, as I invoke them here, represent a much broader process: social bias embedded in technical artifacts, the allure of objectivity without public accountability”

~ Ruha Benjamin @ruha9
“What are the unintended consequences of designing systems at scale on the basis of existing patterns of society?”

~ M.C. Eilish & Danah Boyd, Don’t Believe Every AI You See
@m_c_elish @zephoria
- Word2Vec has known gender and race biases
- It’s in English
- Is it robust to spelling errors?
- How does it perform with malicious data?
Word2Vec has known gender and race biases
It’s in English
Is it robust to spelling errors?
How does it perform with malicious data?
Playing with AI Fairness

Google’s new machine learning diagnostic tool lets users try on five different types of fairness

Posted by David Weinberger, writer-in-residence at FAIR

David is an independent author and currently a writer in residence within Google’s People + AI Research Initiative. During his residency, he will be offering an outside perspective on FAIR researchers’ work; explaining aspects of how AI works; and providing context on AI’s societal meaning beyond the technical sphere. He speaks for himself and the opinions in this post are his.

Researchers and designers in Google’s FAIR (People and AI Research) initiative created the What-If visualization tool as a pragmatic resource for developers of machine learning systems. Using the What-If tool reveals, however, one of the hardest, most complex, and most utterly human, questions raised by artificial intelligence systems: What do users want to count as fair?

---

https://pair-code.github.io

http://aif360.mybluemix.net

https://github.com/fairlearn/fairlearn

https://github.com/jphall663/awesome-machine-learning-interpretability
Expectations

model ➔ API ➔ train & evaluate model ➔ CI/CD ➔ user testing

- Unit Tests

- User testing
choose a useful metric

Evaluate model
Choose threshold
Explain predictions
Fairness Framework
Surprises after deploying the model
Expectations

Bug Triage

bug tracking tool

agile cycle

reproduce, debug, fix, release

user drop off

user testing

logs a bug or submits a complaint
Surprise #5

I can provide marijuana and other drugs which improves health

I want to meet a doctor
Surprise #5

The model has some “bugs”
What is a model “bug”

How to fix the bug?

When is the “bug” fixed?

How do I ensure test regression?

“Bug” priority?
I can provide marijuana and other drugs which improves health

I want to meet a doctor
## Describing the “bugs”

<table>
<thead>
<tr>
<th>Description</th>
<th>Target Description</th>
<th>Prediction</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>I can provide marijuana and other drugs which improves health</td>
<td>I want to meet a doctor</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>I can provide marijuana</td>
<td>I want to meet a doctor</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>I can provide drugs for cancer patients</td>
<td>I want to meet a doctor</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>I can provide general practitioner services</td>
<td>I want to meet a doctor</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>I can provide medicine</td>
<td>I want to meet a drug addiction sponsor</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>I can provide medicine</td>
<td>I want to meet a pharmacist</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>I can provide illegal drugs</td>
<td>I want to meet a drug dealer</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>ID</td>
<td>Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>same-day-problem</td>
<td>If tangibles have the same day but everything is different, they will match when they shouldn't</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>same-place-problem</td>
<td>If tangibles have the same place but everything is different, they will match when they shouldn't</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>for-me</td>
<td>Tangibles with for me in them get less matches</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1-services</td>
<td>If 1 tangible has services, then it experiences a bit negative</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2-services</td>
<td>If both tangibles has service in it, then it experiences a big positive</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>commas</td>
<td>If 1 tangible has a comma in it, then prediction has a big negative</td>
<td></td>
</tr>
</tbody>
</table>
### View match problem

#### Match problems

<table>
<thead>
<tr>
<th>ID</th>
<th>Created at</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2017-02-14</td>
</tr>
</tbody>
</table>

#### Name:

same-day-problem

#### Regular expression:

select * from tangibles where (description ~ 'Monday' or description ~ 'Tuesday' or description ~ 'Wednesday' or description ~ 'Thursday' or description ~ 'Friday' or description ~ 'Saturday' or...

#### Description:

If tangibles have the same day but everything is different, they will match when they shouldn't

### Test patterns

2 positive, 6 negative samples

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide</td>
<td>I can provide iOS development</td>
</tr>
<tr>
<td>Meet</td>
<td>I want to meet mobile app dev</td>
</tr>
</tbody>
</table>

Should match:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide</td>
<td>I can provide social media management and setup in Safety Harbor</td>
</tr>
</tbody>
</table>
Is my “bug” fixed?

![Graph showing classification error over time for different categories: politicians-false-neg, designers-too-general, drugs-doctors-false-pos, tech-too-general.](image)
How do we triage these “bugs”?
How do we triage these “bugs”?

% Users Affected
x
Normalized Error
x
Harm
How do we triage these “bugs”?

<table>
<thead>
<tr>
<th>Problem</th>
<th>Impact Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>the-arts-too-general</td>
<td>2.931529</td>
</tr>
<tr>
<td>health-more-specific</td>
<td>1.53985</td>
</tr>
<tr>
<td>brand-marketing-social-media</td>
<td>1.285735</td>
</tr>
<tr>
<td>developer</td>
<td>1.054248</td>
</tr>
<tr>
<td>1-services</td>
<td>0.960129</td>
</tr>
</tbody>
</table>
Is this new model better than my old model?
Alice replied, rather shyly, “I—I hardly know, sir, just at present—at least I know who I was when I got up this morning, but I think I must have changed several times since then.”
Why is model comparison hard?
Living Test Set

0.8
0.75
Re-evaluate ALL models
Surprise #7

I demoed the model yesterday and it went off-script! What changed?
Surprise #7

Why is the model doing something differently today?
What changed?

❖ My data?
❖ My model?
❖ My preprocessing?
How to figure out what changed?

**Metadata Store**
- experiment: 3
- data: ea2541df
- code: da1341bb
- desc: “Added feature to training pipeline”
- run_on: 10-10-2019
- completed_on: 11-10-2019
- model: model-3
- results: 3

**Model Repository**
- model-3

**Results Repository**
- ea2541df
- da1341bb

**Data Repository**

**Code repository**

**CI/CD**
Expectations

- Prioritization
- bug tracking tool
- reproduce, debug, fix
- logs a bug or submits a complaint
- agile cycle
- user drop off
- user testing
user reports bug → Identify problem → Describe problem with test patterns → Add to model bug tracking tool → Calculate Priority → Triage

- Evaluate model against other models
- Evaluate individual problems
- Select model

“Agile Sprint”

- Retrain

Pick Problem

- Gather More Data for Problem
- Change Model
- Create Features
Surprises

Surprises maintaining and improving the model over time
Expectation

Pick an issue

Generate/select unlabelled patterns

Get them labelled

Add to data set

Retrain
Surprise #8

User behaviour drifts
Now what?

- Regularly sample data from production for training
- Regularly refresh your test set
Surprise #9

Data labellers are rarely experts
The model is not robust
Surprise #10

The model knows when it’s uncertain
Techniques for detecting robustness & uncertainty

- Softmax predictions that are uncertain
- Dropout at Inference
- Add noise to data and see how much output changes
Changing and updating the data so often gets messy
Needed to check the following

- Data Leakage
- Duplicates
- Distributions
Expectation

- Pick an issue
- Generate/select unlabelled patterns
- Get them labelled
- Add to data set
- Retrain
Pick Problem → Generate/select unlabelled data → Get data labelled on crowdsourced platform → Review sample from each data labeller → Escalate conflicting data labels → Expert data label platform → New data!

Data Version Control:
- Merge into dataset
- CI/CD: Runs tests on data

Data Version Control:
- Add to branch of dataset
# The Checklist

**First Release**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Careful metric selection</td>
<td>✔️</td>
</tr>
<tr>
<td>Threshold selection strategy</td>
<td>✔️</td>
</tr>
<tr>
<td>Explain Predictions</td>
<td>✔️</td>
</tr>
<tr>
<td>Fairness Framework</td>
<td>✔️</td>
</tr>
</tbody>
</table>
# The Checklist

**After First Release**

<table>
<thead>
<tr>
<th>Task</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML Problem Tracker</td>
<td>✔️</td>
</tr>
<tr>
<td>Problem Triage Strategy</td>
<td>✔️</td>
</tr>
<tr>
<td>Reproducible Training</td>
<td>✔️</td>
</tr>
<tr>
<td>Comparable Results</td>
<td>✔️</td>
</tr>
<tr>
<td>Result Management</td>
<td>✔️</td>
</tr>
<tr>
<td>Be able to answer why</td>
<td>✔️</td>
</tr>
</tbody>
</table>
## The Checklist

### Long term improvements & maintenance

<table>
<thead>
<tr>
<th>Item</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data refresh strategy</td>
<td>✔️</td>
</tr>
<tr>
<td>Data Version Control</td>
<td>✔️</td>
</tr>
<tr>
<td>CI/CD or Metrics for Data</td>
<td>✔️</td>
</tr>
<tr>
<td>Data Labeller Platform + Strategy</td>
<td>✔️</td>
</tr>
<tr>
<td>Robustness &amp; Uncertainty</td>
<td>✔️</td>
</tr>
</tbody>
</table>
## Things I didn’t cover

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pipelines &amp; Orchestration</strong></td>
<td>Kubeflow, MLFlow</td>
</tr>
<tr>
<td><strong>End-to-end Products</strong></td>
<td>TFX, Sage Maker, Azure ML</td>
</tr>
<tr>
<td><strong>Unit Testing ML systems</strong></td>
<td>“Testing your ML pipelines” by Kristina Georgieva</td>
</tr>
<tr>
<td><strong>Debugging ML models</strong></td>
<td><a href="https://example.com">A field guide to fixing your neural network model by Josh Tobin</a></td>
</tr>
<tr>
<td><strong>Privacy</strong></td>
<td>Google’s Federated Learning</td>
</tr>
<tr>
<td><strong>Hyper parameter optimization</strong></td>
<td>So many!</td>
</tr>
</tbody>
</table>
The End

@alienelf
ja@retrorabbit.co.za
https://retrorabbit.co
https://kalido.me
https://masakhane.io