ShallowVariant
A Child Model for DeepVariant

August Guang, Mary McGrath

December 16, 2019

1 Introduction

DeepVariant [6] is a deep learning model created by the Genomics team at Google Brain to address the biological problem of variant calling, the identification of variants, (i.e. mutations or other evolutionary processes that change the genome) in genome sequencing data. Variant calling is a very important area of research because variants between different populations can be linked to different traits, in particular those that represent disease susceptibility or lead to different treatment outcomes. DeepVariant is a very powerful model that has won contests, but is a very heavy model. Child models train on the predicted labels of the much larger parent model, and despite using much smaller architectures have been shown to produce comparable results [2].

The DeepVariant pipeline consists of three components: `make_examples`, `call_variants`, and `postprocess_variants`. `make_examples` transforms genomic data (an alignment and reads) into 6-channel images. (Figure 1) `call_variants` runs the trained model on the images and outputs 3 probabilities indicating whether a call site is homozygous, heterozygous-alt, or heterozygous. Finally, `postprocess_variants` takes the probabilities and outputs VCF (variant call format) files for use in other bioinformatics applications.

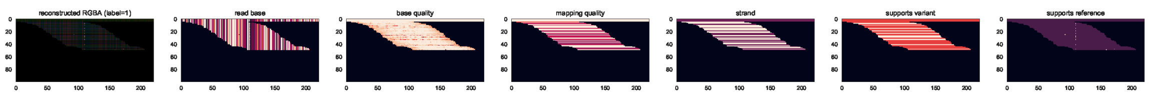

Figure 1: Pileup image example, the first candidate variant in our test dataset. The 6 channels are the read base, base quality, mapping quality, strand, supports variant, or supports reference.

2 Methodology

The ShallowVariant model was trained on part of a single subject of the Genome in a Bottle (GiaB) dataset [7]. (See Discussion for reasoning) The training dataset was randomly sampled from this subject and made up 90% of the total dataset. The test dataset was sampled from a different subject in the GiaB dataset and made up 10% of the total dataset. In total, the dataset used in developing ShallowVariant comprised of 950,000 samples

Data was pre-processed using `make_example`, as it has a `--mode training` option to output the true labels needed for evaluation of ShallowVariant. These images were then used as inputs to `call_variants` to output the 3 probabilities described above. These probabilities were then used as the labels for training ShallowVariant.

3 Results

We developed a child model, ShallowVariant, of DeepVariant's model for predicting variants in genomic data. ShallowVariant's architecture includes a single convolutional layer with 10 3x3 filters using same padding. The output of this layer is then flattened and passed to a dense layer with 3

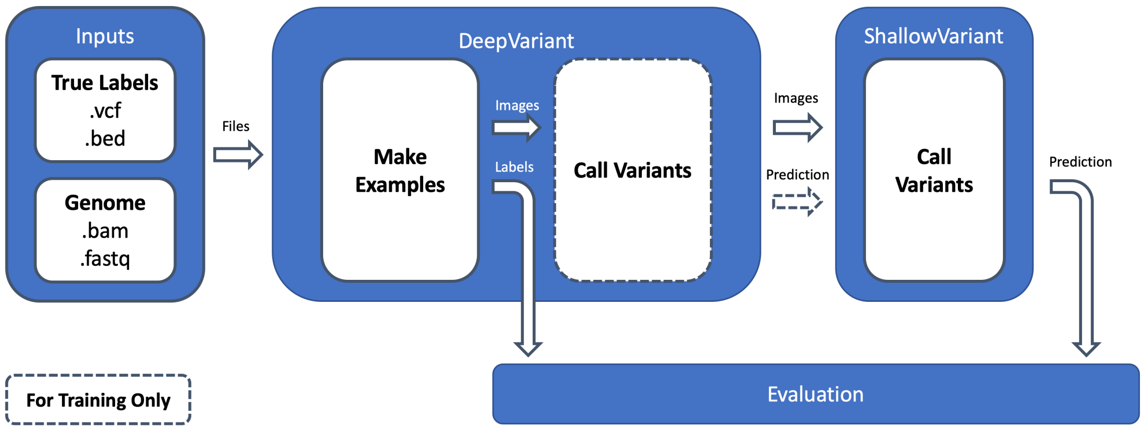

Figure 2: Overview of ShallowVariant training and testing workflow

Model Type	# Filters	Train Accuracy	Test Accuracy
Child	100	96.8%	92.0%
Child	10	96.7%	92.4%
Typical	100	96.8%	88.7%

Table 1: Results of training child model with different number of filters for convolutional layer as well as training a typical model. All models used a learning rate of 0.01 and had the same architecture. Train Accuracy compares the predictions of ShallowVariant to the predictions of DeepVariant, whereas Test Accuracy compares the predictions of ShallowVariant to the ground truth labels.

outputs and a softmax activation. These outputs are the returned probabilities corresponding to whether the variant is homozygous, heterozygous, or heteroyzgous-alt. One epoch of training was completed as that was sufficient for the model converge given the volume of data.

Table 1 shows the results of the tested architectures. These results confirm our expectation that a small model would perform nearly as well as DeepVariant, which has an accuracy of 99.8%, while using a much smaller architecture. It also confirms that the methodology of training on the predictions of DeepVariant instead of the true labels as with a typical model, does perform better. The child model outperforms the typical model by over 4%.

4 Discussion

In total, ShallowVariant (see Figure 3) has 663, 553 trainable parameters whereas DeepVariant has approximately 25 million. DeepVariant was trained using the inception v3 model [5] (see Figure 4) as its base. The much smaller ShallowVariant model also represents a much smaller computational, and therefore environmental footprint. The time to call ShallowVariant on 100,000 samples is approximately 4.5 minutes, a small fraction of the time to do the same on DeepVariant.

4.1 Training data

DeepVariant advertises itself to take 5 hours on a 64-core CPU machine, or about $9.11 on a 16-core CPU machine (no time estimate). However, this is to simply call variants with the trained model on a genome. For our child model, we also needed to generate training data, probabilities, and labels. Generating the training data from a single genome using `make_examples` took over 24 hours, and in fact we decided to kill the process partway through and use what we had, so as to not run out of GCP credits. This meant that we did not train or test on the full set of data that DeepVariant is trained and tested on, so accuracy results are not directly comparable.

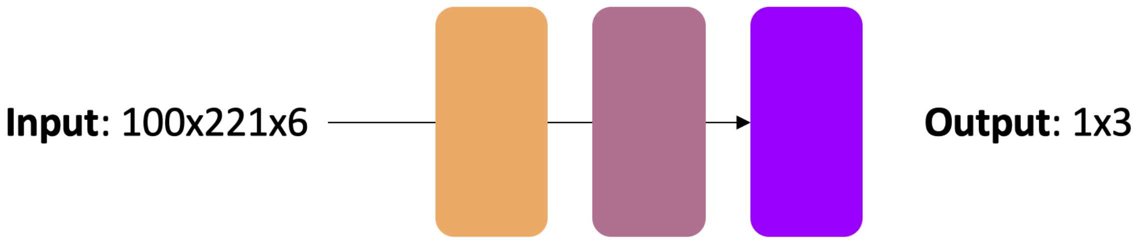

Figure 3: Overview of ShallowVariant architecture, using same format and legend as Figure 4

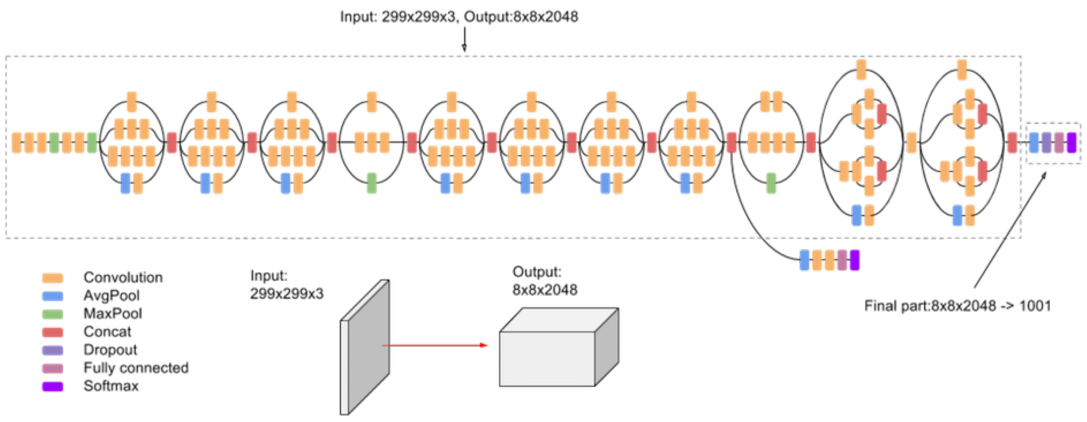

Figure 4: Overview of Inception-v3 architecture, which was used as the DeepVariant base model. Image from https://github.com/tensorflow/models/tree/master/research/inception.

The bigger issue is that we suspect `make_examples` is the bottleneck in the DeepVariant pipeline based on our experience with trying to generate training data. Since our child model still relies on `make_examples` to generate images to feed into the model, it may not provide as much of a speedup as we hoped. Future work could address this by running a side-by-side comparison of the full DeepVariant pipeline and a full ShallowVariant pipeline that includes `make_examples` and `postprocess_variants`, and timing each component.

4.2 Ethical Considerations

While the environmental impacts of this model are positive, those alone are not enough to use the model in practice for human genomes. Variant calling in the human genome is used to make medical decisions such as surgery or prophylactic measures, therefore the error rate of a model must be minimized as much as possible.

An additional consideration is that both DeepVariant and ShallowVariant were developed on only two subjects, one Ashkenazi Jewish and one Chinese. While those genomes provide many variant samples to train and test on, they do not reflect the diversity of genomes in the population at large. GiaB and other genomic datasets are skewed towards samples from people of white, European descent and do not reflect the population of the world as a whole. In fact, a meta-analysis of genomic studies revealed that 78 percent of individuals in such studies were of European descent, 10 percent were Asian, and 2 percent or less for all other groups [4]. This contributes to healthcare inequity as the variants and other genetic markers associated with disease that are discovered are overwhelmingly linked to white European populations, biasing medical diagnoses and care. Meanwhile, variants and genetic markers associated to disease that are more prevalent in other populations remain largely unknown. Unfortunately, this is a complex problem that must be fixed on the data collection, consent and outreach end. Until then one should be aware that all computational genomics tools including DeepVariant and ShallowVariant are likely biased in their predictions for subjects of different races.

5 Challenges & Lessons learned

Our original proposal for this project included further aims to extend this work to *Candida albicans*, a fungus, with a goal to understand if ShallowVariant would extend as well or better to another organism as DeepVariant does (which is not well). However, we ran into challenges with parsing and fetching data at scale which precluded those aims from being included in this phase of the work.

While DeepVariant is implemented using TensorFlow [1] and uses `TFRecords` and `TFExamples`, how the data was stored within those standard formats was custom to DeepVariant and not well documented. To parse the output of `make_examples`, we had to comb through the code of DeepVariant to find where in the code the examples were parsed in its processing of the data. While this was a hurdle, it turned out to be much easier than parsing the output of `call_variants`, which outputted a `TFDataset` with `TFRecords` that contained custom ProtoBuf messages that were built on top of Google's Nucleus library. To parse these messages, we created a modified version of DeepVariant's docker image which loaded in a custom script to use their built package and environment to parse the probabilities within those messages to a text file, which was then usable by ShallowVariant.

Additionally, as discussed in Section 4.1, pulling and pre-processing the data at scale from GiaB was much more computationally, and time intensive than anticipated.

Due to these challenges, we learned:

- a lot about how DeepVariant works
- a lot about the Nucleus library
- a lot about TFRecords and Protobuf objects
- a lot about the difficulties of data pre-processing
- a lot about GCP including gcloud, gsutil, gcsfuse, beta genomics pipelines, and storage buckets

6 Future work

All human variant callers these days have precision and recall above 99% [3]. (`https://precision.fda.gov/challenges/truth/results`) However, these variant callers are optimized for humans, and accuracy drops precipitously when moving to non-human organisms. Although ShallowVariant is relatively accurate and is much faster and more environmentally friendly than the other models, it cannot compete with the other variant callers for humans. One potential way to capitalize on its speed is to use it as a variant caller for non-human organisms instead. We have access to some high-confidence variant calls for *Candida albicans*, a fungus, and could use it 2 ways: (1) as a test dataset to compare ShallowVariant to other models, and (2) partition it to use some as additional training data for ShallowVariant, with the rest being used as test. If accuracy is comparable then ShallowVariant could be a high-speed improvement to existing variant calling tools.

7 Code

The source code used to train and test ShallowVariant can be found at `https://github.com/compbiocore/shallowvariant`.

8 Reflection

As noted in our Challenges, we had hoped to reach further aims in this project, but due to the challenges of parsing and working with the data at scale we only achieved the first aim of our initial proposal. In retrospect, our initial proposal was too ambitious for the time frame. However, we're quite happy with the results we were able to obtain both in regards to the high accuracy of the much simpler model as well as the validation that a child model is significantly better than a typical model trained on true labels.

The division of labor we proposed went almost exactly as planned. Mary built the initial model and code for pre-processing the image and prediction data for training/testing. August pulled the data and determined how to pull and parse the true labels needed for evaluation. Both of us worked on Google Cloud Platform for various parts of the work, though August did the bulk of that work. We ultimately didn't set up an automated parameter sweep due to time constraints, but Mary was able to run the model with a few different configuations. Both of us made the poster and wrote the report.

References

[1] 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. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[3] Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, and Mark A. DePristo. The genome analysis toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, 2010.

[4] Giorgio Sirugo, Scott M. Williams, and Sarah A. Tishkoff. The Missing Diversity in Human Genetic Studies, 2019.

[5] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[6] Amalio Telenti, Christoph Lippert, Pi-Chuan Chang, and Mark DePristo. Deep learning of genomic variation and regulatory network data. *Human molecular genetics*, 27(1):R63–R71, 2018.

[7] Justin M Zook, Brad Chapman, Jason Wang, David Mittelman, Oliver Hofmann, Winston Hide, and Marc Salit. Integrating human sequence data sets provides a resource of benchmark snp and indel genotype calls. *Nature biotechnology*, 32(3):246, 2014.

9 Acknowledgments

Thanks to Paul Stey for the original idea about child models, and to Ashok Ragavendran for feedback on the idea. Thanks to Dylan Sam, our TA mentor, for his feedback on the project.