Анализ сигнала для прогнозирования времени до землетрясения с использованием нейронных сетей

Игнат Легеза

Аннотация

В рамках работы над курсовым проектом было рассмотрено соревнование Kaggle – LANL Earthquake Prediction[1]. Задачей соревнования является прогнозирование времени до землетрясения по данному отрезку сигнала. При решении задачи мы попробовали несколько различных подходов: от простых 1D CNN до обучения эмбеддингов с помощью CPC[2].

1. Введение

На текущий момент нейронные сети широко используются во многих задачах. В частности, невозможно представить решение задач компьютерного зрения без использования сетей. Похожая ситуация наблюдается в некоторых задачах анализа сигналов, однако в данной области сети все же не заменили стандартные подходы классического машинного обучения. Причем иногда применение стандартных подходов оказывается более успешным, нежели применение глубинного обучения.

Главной целью данной работы является изучение применимости методов глубинного обучения в задаче анализа сигнала на примере Kaggle соревнования LANL Earthquake Prediction. Данный формат курсовой работы был выбран по нескольким причинам. Во-первых, есть хорошо подготовленные реальные данные. Во-вторых, можно быстро сравнить результат своего решения с другими на открытой части тестовой выборки. В-третьих, можно генерировать новые идеи, изучая решения других участников.

1.1. Описание задачи

Глобально задача соревнования состоит в следующем: по данной последовательности значений некоторого акустического сигнала нужно спрогнозировать время до землетрясения в последней его точке.

Для моделирования землетрясения в лаборатории используется конструкция, имитирующая движение тектонических плит, подробно описанная (Johnson et al., 2013). Две небольшие пластины из изучаемого материала зажимаются между собой. При этом между ними располагается третья – подвижная пластина, а в промежутках между ней и двумя первыми находится некоторый гранулированный материал. После сбора данной конструкции, на две первые пластины начинают давить с постоянной силой, а третью начинают выталкивать с заданной скоростью. Так происходит до тех пор, пока пластины не соскользнут, что и будет являться срывом (толчком) в рамках лабораторного землетрясения. На протяжении всего эксперимента с подвижной пластины снимаются показания ее ускорения с частотой 4 МГц, которые являются объектами в нашей задаче.

[1] www.kaggle.com/c/LANL-Earthquake-Prediction
[2] Contrastive Predicting Coding (Oord et al., 2018)

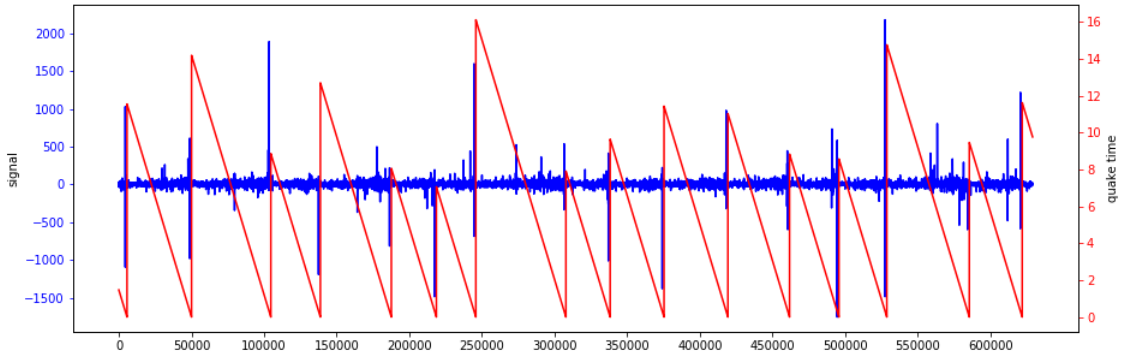

Рисунок 1. Значения сигнала и таргета. Шаг отрисовки 1000 точек.

1.2. Описание данных

Весь тренировочный датасет представляет собой два вектора длинной ≈630 млн. состоящих из значений сигнала и времени до землетрясения (таргет, в секундах). Первый принимает целочисленные значения, а второй вещественные больше ноля.

Одной из немногих проблем с данными можно обозначить тип таргета – float64, что сильно повышает требования к доступной памяти системы. Тем не менее, мы заметили, что потеря точности до float32 не создает каких-либо серьезных проблем, поэтому было решено использовать ее, а не исходную.

Тестовый датасет состоит из 2624 векторов длиной 150 тыс. Для каждого из них нужно получить прогноз таргета к последнему значению сигнала. Метрикой соревнования является средняя абсолютная ошибка (MAE).

Как видно на Рисунке 1 в тренировочном датасете присутствуют всего лишь 16 землетрясений, которые можно определить по обнулениям таргета. Важно отметить, что тренировочные данные были получены в рамках одного эксперимента, чего нельзя сказать о тестовых данных. Более того, о них организаторы не сообщили никаких подробностей. Поэтому нельзя отбрасывать исход, при котором внутри тестового куска могло произойти обнуление таргета. Тем не менее, в рамках работы мы предполагали, что такой подход к генерации теста был бы крайне странным с точки зрения практической применимости итоговой модели.

2. Подходы и используемые модели

В рамках работы были опробованы несколько подходов к решению задачи: от обычной сверточной сети с регрессионной головой до более продвинутого метода с использованием Contrastive Predicting Coding (CPC) (Oord et al., 2018) и классификационной головы на несколько бинов таргета. Рассмотрим все по порядку.

2.1. Регрессия или классификация?

Несмотря на то, что поставленная задача является стандартной задачей регрессии, при ее реализации ”в лоб” возникла серьезная проблема. Используя различные архитектуры с одним вещественным выходом, сетки сходились к одному и тому же тривиальному решению (≈5.85) вне зависимости от входа сети. Одним из вариантов решения данной проблемы может быть использование более хитрой функции потерь. Например, вместо стандартного для задачи регрессии MSE можно использовать MAE или Huber Loss.

Однако, данный трюк не помог, поэтому мы решили преобразовать задачу из регрессии в классификацию на несколько бинов значе-

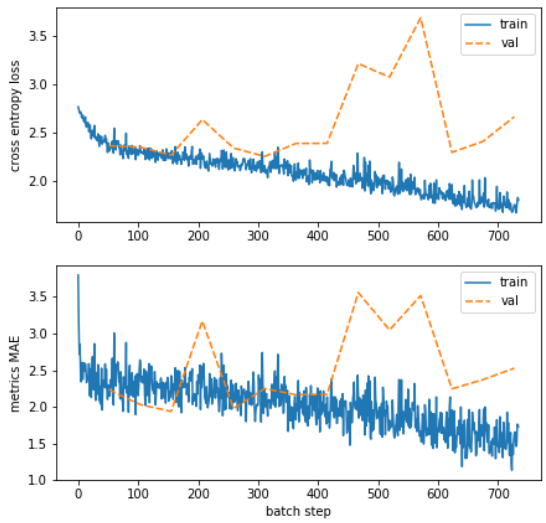

Рисунок 2. Кривые тренировки 1D CNN

ний таргета. В тренировочном датасете таргет принимает значения от 0 до 16.11. Следовательно, мы можем разбить этот отрезок на N меньших отрезков (бинов) и классифицировать вход сети на N непересекающихся классов. После классификации мы брали середину бина, в котором сеть наиболее уверена, и считали ее прогнозом таргета. Мы использовали 16 бинов, так как при идеальной классификации максимальное значение MAE составит 0.5 сек., что, судя по результатам лидерборда, является очень хорошей метрикой. Описанный подход решил проблему сваливания сети к тривиальному решению, поэтому далее при прогнозировании таргета мы использовали только его.

2.2. 1D CNN

В первой итерации мы использовали наиболее простой подход: сверточные сети на исходном сигнале. Инпутом в данном случае являлся батч, в каждом семпле которого содержался отрезок сигнала размером 150 тыс. наблюдений. Подобная логика формирования инпута сети кажется наиболее правильной, так как тестовый датасет состоит из кусков именно такого размера.

Мы попробовали несколько архитектур, по своей сути являющихся немного упрощенными версиями WaveNet (Van Den Oord et al., 2016). Основа используемых архитектур строилась на 1D свертках, в которых с каждым слоем в 2 раза увеличивался размер отступа (dilation) и количество выходных каналов. После слоя свертки использовалась батчевая нормализация (BatchNorm1d) и активация в виде функции ReLU. В конце тела сети применялся адаптивный пулинг (AdaptiveAvgPool1d). Таким образом, в зависимости от модификации архитектуры, на вход классифицирующей голове приходил эмбеддинг размерности от 64 до 256 элементов. Классифицирующая голова в свою очередь состояла из нескольких линейных слоев с активацией ReLU и итоговым выходом в 16 нейронов. В качестве функции потерь в данной архитектуре мы использовали стандартный для задачи классификации лосс – кросс-энтропию.

На Рисунке 2 показаны кривые тренировки описанной архитектуры. На них хорошо видно, что модель начала переобучаться с шестой эпохи. Использование различных методов регуляризации, например: дропаут и L2, не показало хороших результатов. Несмотря на кажущуюся хорошую метрику на валидации, на тестовых данных модель показала себя не сильно лучше той, которая сходилась к тривиальному решению.

2.3. CNN на спектрограммах

Во второй итерации мы попробовали преобразовывать входной сигнал в спектрограмму и работать с ней как с картинкой. Инпут в данном случае имел те же размерности как в предыдущем примере за одним исключением. Перед подачей в сеть каждый семпл в батче преобразовывался из вектора в матрицу, представляющую собой спектрограмму входного сигнала. Данное преобразование производилось с помощью функ-

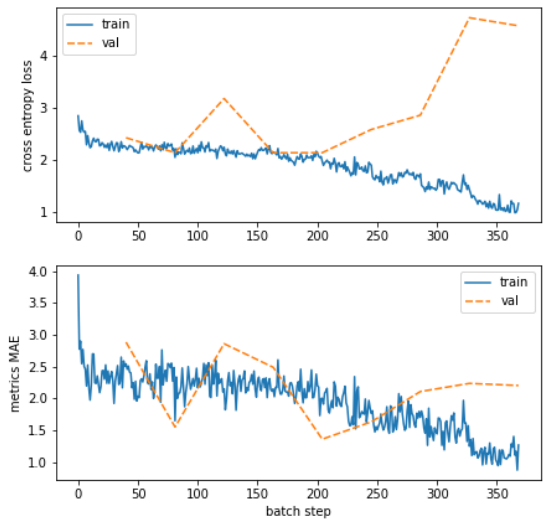

Рисунок 3. Кривые тренировки ResNet-34 на спектрограммах

ции spectrogram из пакета scipy. На получившихся спектрограммах мы попробовали стандартные для работы с картинками архитектуры сетей. В частности использовались предобученные на ImageNet ResNet34 и ResNet50 (He et al., 2016).

Также мы протестировали различные гиперпараметры функции, преобразующей сигнал в спектрограмму: размеры окна и перекрытия, тип оконной функции. Лучший результат показал размер окна равный 512 наблюдениям с перекрытием 64 и последующей постобработкой функцией Ханна. На Рисунке 3, также как и с предыдущей архитектурой, мы наблюдаем сильное переобучение при лучшем значении метрики на валидации, что тем не менее не отразилось при прогнозировании на тестовом датасете.

2.4. Обучение представлений через CPC

В третьей итерации мы решили начать с обучения хороших эмбеддингов, с последующим их использованием в роли инпутов классифицирующей сети. Для обучения эмбеддингов мы применили подход CPC, описанный (Oord et al., 2018).

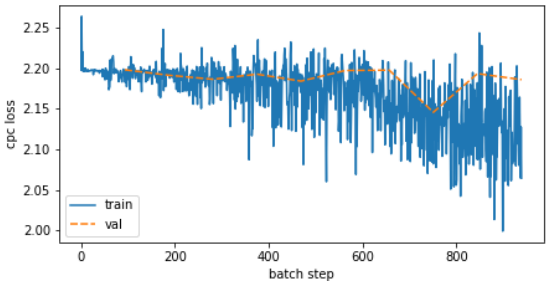

Рисунок 4. Кривые тренировки CPC сети

Идея данного подхода заключается в следующем. Мы используем две сети: первая является энкодером, который учит скрытые представления, а вторая – авторегрессионной моделью, которая должна по предыдущим выходам энкодера отличить его следующий выход (позитивный пример) от каких-либо других (негативные примеры). В виде энкодера мы использовали сеть из нескольких слоев с 1D сверткой, батчевой нормализацией и ReLU. Авторегрессионной моделью в нашем случае являлась рекуррентная сеть GRU. Лоссом архитектуры CPC является InfoNCE, суть которого заключается в SoftMax преобразовании выхода сети по негативам и позитивам с последующим усреднением по батчу и умножением на -1.

Как показано на Рисунке 4, обучение сети при описанном подходе проходило крайне медленно. После десятка эпох обучения CPC мы добавили классифицирующую голову, принимающую на вход аутпуты GRU. Результатом такого трюка стало моментальное переобучение классифицирующей головы. При этом остальная часть сети не показывала переобучения, однако продолжать обучение также не хотела. Также мы пробовали добавлять классифицирующую голову сразу на старте обучения, но такой подход приводил к тому, что голова быстро ломала всю предшествующую часть сети, градиенты которой обращались в ноль.

3. Выводы

Несмотря на то, что результаты оказались не лучшими, нами были изучены и опробованы различные подходы к решению задач анализа сигналов. В частности, были рассмотрены нетривиальные и достаточно свежие подходы, например, CPC.

Также стоит отметить, что наилучшие результаты в данном соревновании были получены с помощью классических методов. Таких как ручная генерация фичей, с последующим обучением моделей градиентного бустинга на деревьях. Среди решений с хорошим результатом есть несколько примеров использования нейронных сетей, однако это были достаточно примитивные сверточные архитектуры на фичах, сгенерированных вручную.

Список литературы

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

Johnson, P. A., Ferdowsi, B., Kaproth, B. M., Scuderi, M., Griffa, M., Carmeliet, J., Guyer, R. A., Le Bas, P.-Y., Trugman, D. T., and Marone, C. Acoustic emission and microslip precursors to stick-slip failure in sheared granular material. Geophysical Research Letters, 40(21):5627–5631, 2013.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018.

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. SSW, 125, 2016.