Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)
Факультет управления и прикладной математики
Кафедра «Интеллектуальные системы»
при Вычислительном центре им. А. А. Дородницына РАН

Черных Владимир Юрьевич

Оптимизация сложности моделей глубокого обучения в условиях нехватки данных

010900 — Прикладные математика и физика

Бакалаврская диссертация

Научный руководитель:
к.ф.-м.н
Рябенко Евгений Алексеевич

Москва
2016 г.

Содержание

Аннотация

В данной работе рассматривается задача распознавания движений человека по сигналам электроэнцефалографии. Для ее решения предлагается использовать глубинные сверточные нейронные сети (Convolutional Neural Networks). В статье описана разработанная структура сети и приводятся данные вычислительного эксперимента, подтверждающие, что использование сверточных нейронных сетей является оправданным и дает результат, сравнимый с классическими методами анализа данных ЭЭГ.

1 Введение

2 Постановка задачи и используемые обозначения

Задачу оптимизации сложности моделей глубокого обучения условно можно разбить на две подзадачи. Первая — это правильно выбрать структуру нейронной сети. Вторая — при выбранной и зафиксированной структуре обучить модель. Здесь будут описаны оба этапа.

В качестве входных данных есть конечная выборка $\mathfrak{D} = \{(\mathrm{X}_i, \mathbf{y}_i)\}_{i=1}^{\mathrm{N}}$, где $\mathrm{X}_i \in \mathcal{X} = \mathbb{R}^n \times \mathbb{R}^c$ — пространство c-канальных временных рядов длины n, $\mathbf{y}_i \in \mathcal{Y}$ — пространство меток классов. Таким образом, признаковое описание одного объекта есть матрица размера n × c. Классы представляются с помощью унитарной (one-hot) кодировки — рассматривается k-мерное (где k - число классов) пространство $\mathcal{Y} = \{\mathbf{y} \in [0;1]^k\}$. Каждая компонента y_i вектора $\mathbf{y}$ дает вероятность принадлежности объекта к i-ому классу. На выборке $\mathfrak{D}$ метки известны достоверно, поэтому в ней все векторы ответов будут иметь бинарный вид $\{0;1\}^k$. Заметим, что мы не делаем ограничение на количество классов, которым принадлежит объект, т.е. в зависимости от специфики выборки могут быть рассмотрены как случаи мультиклассовой (multi-class) классификации, так и классификации с многими метками (multi-label). Произведем разбиение выборки на обучающую и валидационную $\mathfrak{D} = \mathfrak{D}_l \cup \mathfrak{D}_v$ размерами N_l и N_v соответственно.

2.1 Обучение модели

Введем априорно множество классификаторов $\mathcal{F} = \{f : \mathcal{X} \mapsto \mathcal{Y}\}$, с которыми мы работаем. Отметим, что $\mathcal{F}$ можно отождествить с пространством $\mathcal{W}$ параметров модели. Для нейронных сетей при зафиксированной структуре модели $\mathcal{W}$ будет пространством весов сети. Далее обозначения f и $\mathbf{w}$ будут взаимозаменяемыми. Предположим, что есть идеальная целевая зависимость $f^* : \mathcal{X} \mapsto \mathcal{Y}$. Требуется выбрать классификатор $f^*_{\mathcal{F}} \in \mathcal{F}$, наилучшим образом приближающий f^* [1].

Для формализации понятия наилучшего приближения введем функцию ошибки $\mathscr{L}_{\mathrm{e}}(\mathbf{y}, \hat{\mathbf{y}})$. Величина, которую мы всегда хотим минимизировать — точность классификации (accuracy). Но реально это сделать не получается, т.к. функция потерь не является выпуклой. Поэтому на практике используется ее приближение, которое да-

лее будет обозначено как $\mathscr{L}(\mathbf{y}, \hat{\mathbf{y}})$. Задача записывается в виде:

$$f^*_{\mathcal{F}} = \arg\min_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{(\mathrm{X},\mathbf{y})} \left[\mathscr{L}_{\mathrm{e}}(\mathbf{y}, f(\mathrm{X}))\right] = \arg\min_{f \in \mathcal{F}} \mathrm{L}(f).$$

Этот подход называется минимизацией риска. Но так как реальное распределение данных в пространстве $\mathcal{X} \times \mathcal{Y}$ нам неизвестно, то будем минимизировать эмпирический риск:

$$\hat{f}_{\mathrm{e}} = \arg\min_{f \in \mathcal{F}} \frac{1}{N_l} \sum_{\mathfrak{D}_l} \mathscr{L}_{\mathrm{e}}(\mathbf{y}_i, f(\mathrm{X}_i)) = \arg\min_{f \in \mathcal{F}} \mathrm{L}_n(f).$$

Теперь сделаем финальное уточнение. Так как оптимизировать точность классификации напрямую мы не можем в силу отсутствия выпуклости, то будем оптимизировать ее оценку сверху $\mathscr{L}(\mathbf{y}, \hat{\mathbf{y}})$, выбор которой зависит от конкретных условий. Итоговая формулировка задачи, которая будет решаться при выбранной структуре модели:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{N_l} \sum_{\mathfrak{D}_l} \mathscr{L}(\mathbf{y}_i, f(\mathrm{X}_i)) = \arg\min_{\mathbf{w} \in \mathcal{W}} Q(\mathbf{w}, \mathfrak{D}_l, \mathscr{L}).$$

2.2 Выбор модели

Решая сформулированную выше в разделе 2.1 задачу, мы находим с некоторой погрешностью лучший классификатор $f^*_{\mathcal{F}}$ из класса $\mathcal{F}$. Погрешность дает потерю в качестве $\mathrm{L}(\hat{f}) - \mathrm{L}(f^*_{\mathcal{F}})$. Это ошибка оценивания. Вспомним, что $\mathcal{F}$ был задан априорно. Будем выбирать еще и оптимальный класс. Иными словами, теперь структура нейронной сети не фиксирована, т.е. можно варьировать различные гиперпараметры модели, как то: количество слоев и их тип, число нейронов в каждом слое, активационные функции и многие другие. Рассмотрим пространство гиперпараметров $\mathcal{H}$. Каждая точка $\mathbf{h} \in \mathcal{H}$ определяет структуру модели. Зададим параметрическое множество семейств классификаторов $\{\mathcal{F}_{\mathbf{h}}\}_{\mathbf{h} \in \mathcal{H}}$. Для выбора надо уметь сравнивать два семейства $\mathcal{F}_{\mathbf{i}}$ и $\mathcal{F}_{\mathbf{j}}$. Логично выбирать то семейство, у которого функция наилучшего приближения имеет меньший риск, т.е. $\mathcal{F}_{\mathbf{i}}$ лучше $\mathcal{F}_{\mathbf{j}}$ если $\mathrm{L}(\hat{f}_{\mathcal{F}_{\mathbf{i}}}) < \mathrm{L}(\hat{f}_{\mathcal{F}_{\mathbf{j}}})$.

Опишем два подхода к решению проблемы неизвестности распределения в пространстве $\mathcal{X} \times \mathcal{Y}$.

2.2.1 SRM

Первый называется структурная минимизация эмпирического риска (SRM) [2, 3, 4]. В силу невозможности сравнить чистые риски, будем сравнивать их эмпирические

аналоги, из-за чего вводится ненулевой порог сравнения: $\mathcal{F}_{\mathbf{i}}$ предпочтительнее $\mathcal{F}_{\mathbf{j}}$ при $Q(\hat{f}_{\mathcal{F}_{\mathbf{i}}}) - Q(\hat{f}_{\mathcal{F}_{\mathbf{j}}}) < \alpha(\mathbf{i}, \mathbf{j}, N)$. Вводя порог как разность штрафов каждой модели за сложность $\alpha(\mathbf{i}, \mathbf{j}, N) = p(\mathbf{j}, N) - p(\mathbf{i}, N)$, получаем следующую формулировку задачи выбора модели с регуляризацией:

$$\hat{\mathbf{h}} = \arg\min_{\mathbf{h} \in \mathcal{H}} \left(Q(\hat{f}_{\mathcal{F}_{\mathbf{h}}}) + p(\mathbf{h}, N) \right).$$

Далее, решение и получаемая ошибка аппроксимации $\mathrm{L}(f^*_{\mathcal{F}_{\hat{\mathbf{h}}}}) - \mathrm{L}(f^*)$ зависят от выбора штрафной функции p, но обычно итоговое минимизируемое выражение имеет единственный минимум.

2.2.2 Кросс-валидация

Другим методом выбора модели, чаще применяемом на практике, является кросс-валидация [5]. Отличие от SRM состоит в том, что от сравнений рисков $\mathrm{L}(\hat{f}_{\mathcal{F}_{\mathbf{i}}})$ и $\mathrm{L}(\hat{f}_{\mathcal{F}_{\mathbf{j}}})$ мы переходим не к сравнению эмпирических рисков, но к сравнению других оценок риска. При этом порог сравнения, т.е. функцию α, мы полагаем равной нулю — поэтому не появляется штрафной член. Ниже описан сам метод оценивания риска. Обучающая выборка дробится на q непересекающихся блоков одинаковой длины $\mathfrak{D}_l = \bigcup_{i=1}^{q} \mathfrak{D}_l^i$. Каждый блок по очереди становится контрольной подвыборкой, при этом обучение производится по остальным $q - 1$ блокам. Тогда итоговое семейство классификаторов выбирается согласно следующему правилу:

$$\hat{\mathbf{h}} = \arg\min_{\mathbf{h} \in \mathcal{H}} \frac{1}{q} \sum_{i=1}^{q} Q(f_{\mathrm{CV}}, \mathfrak{D}_l^i, \mathscr{L}),$$

$$f_{\mathrm{CV}} = \arg\min_{f \in \mathcal{F}_{\mathbf{h}}} Q(f, \mathfrak{D}_l \setminus \mathfrak{D}_l^i, \mathscr{L}).$$

Это наиболее часто используемый на практике метод, поскольку он напрямую оценивает обобщающую способность и довольно прост в реализации. SRM сложен тем, что требует вложенности классов по сложности, вычисляемой через VC-размерность, друг в друга [2]. VC-размерность, в свою очередь, может быть просто вычислена только для очень простых классов функций $\mathcal{F}$, например линейных. В случае же нейронных сетей на настоящий момент существуют только очень неточные верхние оценки [6, 7]. Поэтому в дальнейшем в работе будет применяться кросс-валидация.

3 Сверточные нейронные сети

Нейронные сети являются универсальным методом машинного обучения и хорошо зарекомендовали себя во множестве задач. Рассмотрим один из типов моделей, входящий в этот широкий класс, а именно сверточные нейросети. В отличии от обычных нейронных сетей [8], в сверточных сетях [9, 10] различают несколько типов слоев в зависимости от их свойств и назначения. Приведем краткое описание каждого них.

- Сверточные

  Название слоя произошло от названия математической операции свертки. Пусть есть сигнал $x(t)$ и функция взвешивания $w(t)$, которую еще называют ядро или фильтр. Тогда операция свертки запишется как $s(t) = (x * w)(t) = \int x(a)w(t-a)da$. Переходя к терминам сверточных нейронных сетей, x — вход слоя, обозначаемый далее X, w — фильтр, обозначаемый за K. Рассматривая дискретный случай и ядро с ограниченным носителем, получим $\mathrm{S}(i) = \sum_m \mathrm{X}(i)\mathrm{K}(i-m)$. В большинстве библиотек машинного обучения реализуется чуть другая операция — кросс-корреляция:

  $$\mathrm{S}(i) = \sum_m \mathrm{X}(i+m)\mathrm{K}(m),$$

  или в двухмерном случае:

  $$\mathrm{S}(i,j) = \sum_m \sum_n \mathrm{X}(i+m, j+n)\mathrm{K}(m,n).$$

  Таким образом, основное отличие сверточных слоев от полносвязных состоит в том, что каждый нейрон в нем соединен только с ограниченным числом нейронов из предыдущего слоя, а не со всеми. Обычно такой слой рассматривают как совокупность фильтров. Каждому фильтру ставят в соответствие несколько параметров (рис. 1):

  1. Размер (filter size)
  2. Шаг (stride)
  3. Набор весов (weights and bias)

  Каждый фильтр в сети реализуется набором нейронов, каждый из которых подсоединен только к своей области видимости — это свойство носит название

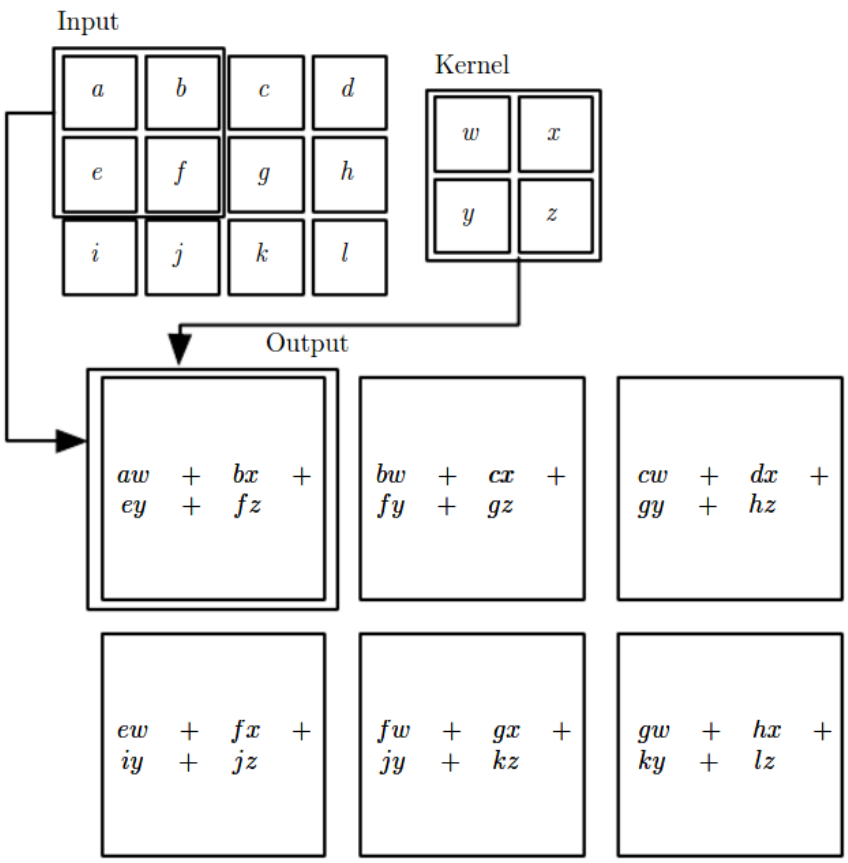


Рис. 1: Свертка одним фильтром

локальности зоны восприимчивости нейрона (local receptive field). Еще одной важной парадигмой в этой теории является разделение параметров (parameters sharing) — идея о том, что нейроны, реализующие один фильтр, имеют одинаковые веса. Объединяя два этих концепта в одно, можно сказать, что сверточный слой обладает свойством разреженности связей (sparse connectivity). В силу того, что размер фильтра обычно на порядки меньше размера входа, требуется гораздо меньше параметров и машинного времени для того, чтобы вычислить выход каждого нейрона.

Главным назначением одного конкретного сверточного слоя является выделение простых паттернов во входе с помощью обучаемых фильтров. Располагая сверточные слои один за другим и в комбинации с другими типами слоев, получаем, что с ростом глубины сети растет абстрактность и сложность выделяемых признаков. Так если фильтры настроены на выделение прямых линий на картинке, то после первого слоя будут распознаны только прямые линии, после второго — их комбинаций, то есть, например, выпуклые фигуры и так далее.

- Активационные

  Выход сверточного слоя есть линейной преобразование над его входом. Композиция линейных преобразований дает линейное преобразование. У нас же есть желание аппроксимировать функции высокой сложности с помощью нейронной сети. Поэтому необходимо добавлять нелинейности. Именно это и делает активационный слой. Обычно в качестве нелинейных функций в промежуточных активационных слоях используют ReLU [11] и ее параметрические аналоги такие как LeakyReLU или PReLU [12]. Использование традиционных функций активации, таких как сигмоида или гиперболический тангенс, несет за собой проблемы размывающихся (vanishing) или взрывающихся (exploding) градиентов на этапе обучения, а также некоторые другие [11].

- Пулинг

  В самом общем виде пулинг — замена элементов входа на некоторую статистику его близлежащих элементов. Главная функция таких слоев — снижение размера представления данных после очередного слоя, чтобы уменьшить число параметров, количество вычислений в сети и избежать переобучения. Наиболее часто используемый вид статистики — максимум (рис. 2), когда оператор возвращает максимум из области, поданной на вход. Также реже используются другие функции — среднее по области; L2-норма; взвешенное среднее с весами, пропорциональными расстоянию до центральной точки и другие. Выбор конкретной функции зависит от специфики задачи.

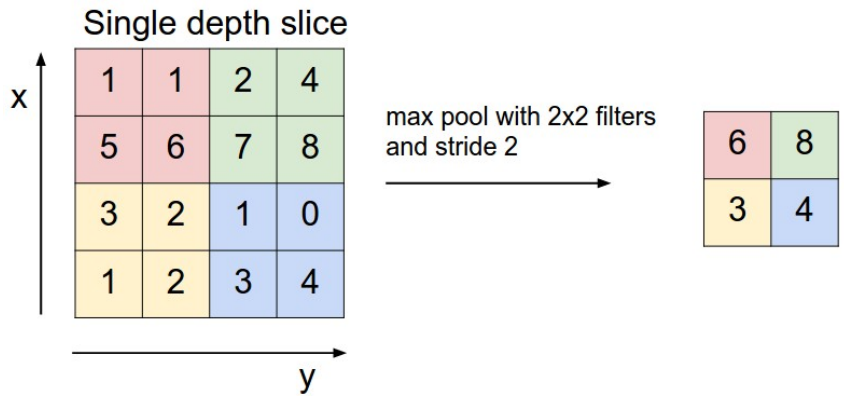


Рис. 2: max-пулинг на квадратной области

Наряду со снижением размера данных, пулинг-слои выполняет еще одну очень важную функцию. Они делают представление данных приближенно инвариант-

ным относительно малых трансляций признака во входных данных. Это легко объяснить — при маленьком сдвиге во входных данных значения на выходе пулинг-слоя почти не будут меняться, т.к. происходит агрегация с помощью статистики по определенной окрестности. Это свойство может быть полезно, если мы больше интересуемся наличием признака во входе, нежели точным его положением. Чтобы сделать итоговое представление инвариантным не только к трансляциям, но и к другим преобразованиям, например вращениям, можно агрегировать в пулинг-слое выходы из сверточных слоев параметризованных по-разному, т.е. имеющих различный формы и веса фильтров.

- Полносвязные

  Все предыдущие слои, скомбинированные в некоторую структуру, по сути своей являются генератором нового представления входных данных. По полученным признакам, используя обыкновенную нейронную сеть из нескольких полносвязных слоев, производится классификация.

- Дополнительные

  Архитектуры почти всех сверточных нейронных сетей содержат в себе слои, описанные выше. Но глубокие нейронные сети — это модели с очень большим числом параметров, как правило порядка 10^6 и более. Поэтому в целях уменьшения переобучения используются некоторые дополнительные слои, например Dropout, DropConnection, нормализационные, шумовые и множество других [13]

Отметим также, что наиболее часто используемая архитектура сверточных нейронных сетей выглядит следующим образом: стекинг нескольких сверточных слоев друг за другом, затем, как правило, нелинейная функция активации и пулинг-слой. Такая последовательность может повторяться несколько раз. Более подробное описание современных архитектур можно найти в [14, 15, 16]

4 Методы анализа ЭЭГ

4.1 Традиционные техники

Классический подход к решению задачи анализа данных ЭЭГ являет собой многоступенчатую систему [17, 18], в которой можно выделить следующие основные эта-

пы:

1. Частотная фильтрация.

   В сигнале ЭЭГ существует множество помех, которые условно можно разделить на две категории — физиологические и технические. Частотная фильтрация, например полосовая, FIR, IIR и другие, может быть использована [19] для того, чтобы исключить из сигнала разного рода технические помехи, как то: наводки от линии электропитания, моментные изменения сопротивления электродов в датчиках и т.д.

2. Удаление артефактов.

   Этот этап связан с фильтрацией данных от физиологических артефактов, то есть сигналов, не относящихся непосредственно к деятельности мозга — мускульная активность, движение и моргание глазами и т.д. Эти помехи могут быть исключены несколькими путями, например ручным выбором признаков и отбросом ненужных каналов на основе визуального анализа; использованием метода независимых компонент ICA [20] с разложением исходного сигнала в виде $X = SA$, где S — независимые компоненты, нахождением артефактных компонент в S, обнулением соответствующих коэффициентов в $A \to \tilde{A}$ и последующем восстановлением сигнала $\tilde{X} = S\tilde{A} = XA^{-1}\tilde{A}$ или же любым другим методом.

3. Уменьшение размерности.

   Показания ЭЭГ с разных, но близких каналов сильно коррелированы. Чтобы избавиться от мультиколлинеарности можно воспользоваться одним из многих методов понижения размерности, например методом главных компонент PCA или методом общих пространственных структур CSP [21], который обычно используют в задачах ЭЭГ.

4. Устранение зависимости от субъекта.

   Показания разных людей и даже показания одного и того же человека в разные сессии имеют большую вариабельность (***ссылка на что-нибудь медицинское***), следовательно, необходимо искать стационарные подпространства и признаки, которые будут устойчивы относительно этой вариабельности. Одним

из возможных способов является усовершенствованный алгоритм CSP, использующий дивергенцию Кульбака-Лейблера [22].

5. Разбиение сигнала на отрезки и последующая классификация.

   После предобработки сигнала производится классификация по разбитым отрезкам. Можно использовать любой алгоритм, но чаще используются линейные модели — SVM, логистическая регрессия, или простые нелинейные алгоритмы — kNN, полносвязные нейронные сети [17].

Таким образом, традиционный метод анализа требует человеческого участия и состоит из нескольких шагов, каждый из которых использует свои алгоритмы и, как следствие, свои критерии качества, что также является недостатком.

4.2 Нейросетевой подход

Стоит отметить, что первые три шага классического анализа есть линейные преобразования исходного сигнала, поэтому можно попробовать выполнять их с помощью одного и того же фреймворка — сверточной нейронной сети. Сверточные слои, как обсуждалось ранее в разделе 3, в силу локальной зоны видимости нейронов и линейной функции активации, по сути своей являются инструментом автоматического подбора фильтров из классического анализа сигналов. Пулинг-слои, в свою очередь, обеспечивают трансляционную инвариантность отыскиваемых паттернов. Таким образом, учитывая тот факт, что нейронные сети хорошо себя зарекомендовали в решении задач анализа сигналов и изображений [14], логичным и перспективным выглядит их применение к анализу ЭЭГ.

Одна из главных проблем в области глубоких нейронных сетей — правильный подбор архитектуры сети, т.е. класса $\mathcal{F}$. Сейчас это делается либо на основе неформализуемых экспертных знаний в области, либо по опыту предыдущих задач, либо методом проб и ошибок, т.е. по сути наугад.

В данной работе предлагается использовать кросс-валидацию (см. раздел 2.2.2) для выбора локально оптимальной модели по данным. Пространство структур моделей $\mathcal{H}$ зададим с помощью описанных ниже параметров сверточной нейросети.

- Количество нейронов в полносвязных классификационных слоях

  Самое большое число параметров сети сосредоточено именно в полносвязных

слоях по понятным причинам. Именно поэтому переобучение наиболее вероятно в этих слоях и поэтому там имеет смысл варьировать число нейронов.

- Вероятность Dropout в слоях классификации

  Dropout позволяет избежать переобучения путем выкидывания каждого нейрона в слое с вероятностью p на этапе обучения [23]. Это не дает весам сильно уходить в разброс, что было бы явным признаком переобученности.

- Длина временного ряда n в каждом из объектов

- Параметры фильтров в первом сверточном слое в зависимости от длины временного ряда n

  Это необходимо для того, чтобы размер представления данных после первого слоя не сильно изменялся от длины временного ряда n.

Также в работе рассматривается проблема переноса архитектуры сети с одних больших данных ЭЭГ на другие маленькие. Предлагается использовать следующую стратегию:

- Берем хорошую, но сложную модель на больших данных
- Постепенно уменьшаем объем больших данных, пока он примерно не сравняется с объемом маленькой выборки
- На каждом шаге выбираем лучшую архитектуру из пространства $\mathcal{H}$

В идеальном случае должен наблюдаться следующий вариант — начальная модель слишком сложна, переобучается и дает плохое качество; наивная модель слишком проста, чтобы выделить все закономерности из данных; максимум качества находится где-то посередине.

В данной работе проверяется несколько гипотез о переносимости архитектуры сети.

Гипотеза 4.1. *Модель без изменений можно перенести на другие данные без значимой потери в качестве.*

Забегая вперед, отметим, что эта гипотеза не подтверждается, что будет показано в разделе 6, и поэтому рассматриваются еще две гипотезы о природе плохой переносимости моделей.

Гипотеза 4.2. *Модель плохо переносима из-за различия в объеме данных.*

Гипотеза 4.3. *Плохая переносимость обусловлена излишней сложностью модели.*

Проверка этих гипотез и вычислительный эксперимент приведены в разделе 6, но прежде опишем данные, которые используются в работе.

5 Описание данных

В работе исследуется 4 различных набора данных. Краткое описание каждого датасета приведено в таблице 1.

Структура		Непрерывная		Попытками	
Название		Kaggle EEG	BBCI IV-I	Ewan Nurse	BBCI II-IV
Год		2014	2007	2015	2002
Размер	**Попыток**	~ 250	~ 400	~ 1350	316
	Точек	$\sim 1.5 \cdot 10\hat{}6$	$\sim 2 \cdot 10\hat{}6$	$\sim 1350 \cdot 100$	$316 \cdot 50$
Частота		500	1000	250	100
Каналов		32	59	10 – 62	28
Классов		6	3	3	2

Таблица 1: Краткое описание данных

В настоящее время различают две основных структуры ЭЭГ–датасетов — непрерывная и попытками (trials). В непрерывном случае снятие показаний с датчиков ведется потоково, в то время как субъект выполняет заданный набор команд. Этот вариант наиболее приближен к реальным задачам BCI, в которых в большинстве своем требуется онлайн распознавание действий или намерений субъекта по данным ЭЭГ. Подход с попытками подразумевает изначальную экспертную (или любую другую, но уже заданную) нарезку временного ряда на отрезки, внутри которых мозг находится в определенном ментальном состоянии, например человек выполняет конкретное моторное действие. Такие отрезки имеют, как правило, равную длину. Их наличие упрощает решение задачи, но отдаляет на один шаг от реальных задач, где такое разбиение не дается и надо встраивать его в сам алгоритм.

Проблема нарезки непрерывных данных решается в этой работе следующим образом — используется скользящее окно определенного размера n. Для каждой точки

во времени обучающим объектом является значение n точек ряда перед текущей точкой по всем c каналам, а обучающей меткой — метка действия в этой временной точке.

Отметим еще одну особенность в работе с непрерывными данными. В ходе решения задачи переноса модели требуется уменьшать объем данных. Если в случае датасетами, которые имеют структуру попыток все ясно — просто убираем из выборки попытки таким образом, чтобы баланс классов примерно сохранялся в исходном состоянии, то с непрерывными данными не все так очевидно. Заметим, что в обоих используемых непрерывных датасетах есть ярко выраженная структура — период выполнения действия сменяется таким же или бóльшим по промежутку периодом бездействия. Поэтому по размеченным данным удается с очень хорошей точностью нарезать данные по попыткам. В классическом анализе временных рядов данные всегда отсекаются с одного из концов ряда — это обусловлено тем, что вырезая данные из середины можно нарушить их структуру и тем самым потерять информацию. Во время уменьшения объема данных будем удалять их с начала ряда, отрезая размеченные до этого попытки. Это связано с тем, что с конца ряда данные берутся для валидации и выкидывая данные с конца обучающей выборки мы бы столкнулись с только что описанной проблемой.

Теперь рассмотрим каждый из наборов данных подробнее.

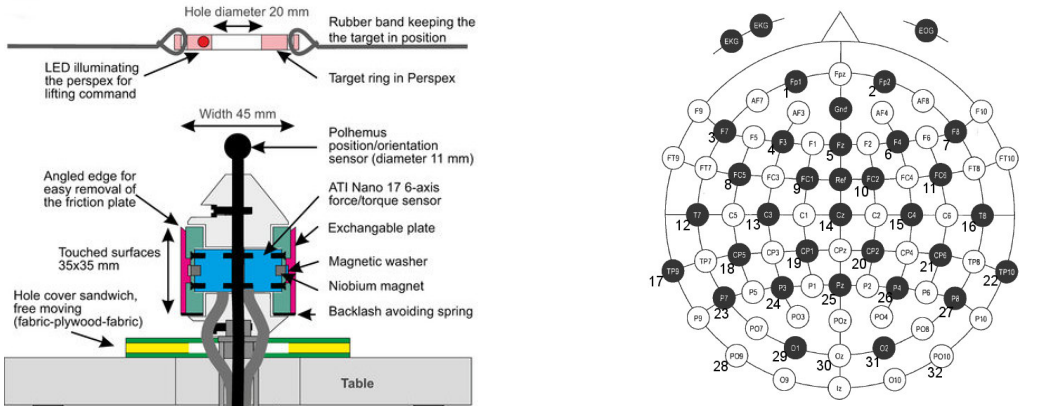
- Kaggle EEG [24]

  Данные имеют непрерывную структуру и содержат записи ЭЭГ людей, которые выполняли одну и ту же последовательность из 6 действий [25]:

  1. Начало движения
  2. Первое касание объекта
  3. Захват с помощью большого и указательного пальцев
  4. Поднятие и удержание в течение нескольких секунд
  5. Помещение объекта в исходное положение
  6. Разжатие пальцев

  Этот датасет предназначен для определения ощущений, намерений и действий основываясь на данных ЭЭГ. 12 участников выполняли описанную последовательность действий с предметами, различающимися по весам (165, 330 или

660 грамм) и типу поверхности (наждачная бумага, замша или шёлк), причем субъект не был уведомлен о данных изменениях, что заставляло регулировать силу сжатия по ходу попытки. Стимулом к началу попытки служил световой сигнал LED-дисплея. Данные снимались одновременно с помощью 32 датчиков на специально собранной для эксперимента установке (см. рисунки 3a, 3b). Для каждого из 12 субъектов было проведено 10 серий измерений, в каждой из которых было примерно по 30 попыток. В качестве выборки берутся первые 8 серий для каждого субъекта, а 9 и 10 серии оставляются для тестирования.



(a) Схема установки

(b) Расположение электродов

Рис. 3: Описание установки Kaggle EEG

- Berlin BCI IV-I [26]

  Выборка также имеет непрерывную структуру, но она содержит данные исследований задачи “воображаемого движения” (motor imagery). Отличие от реальных движений состоит в том, что человек лишь представляет, что он делает движение определенной конечностью, но по факту его не совершает. Такие задачи актуальны в связи с разработкой методов протезирования конечностей и управления ими с помощью BCI [27]. Конкретно в этом эксперименте каждому человеку было предложено выбрать два варианта движений из трех — левая рука, правая рука, любая нога. Таким образом получается три конечных класса — два действия и состояние покоя. Сигналом к началу попытки на обучающей выборке служил визуальный стимул в виде стрелки в нужную сторону. На тестовой же выборке это был звуковой сигнал, что осложняет задачу.

- Ewan Nurse [28]

Этот набор данных, в отличие от двух предыдущих, имеет структуру попыток. Дизайн эксперимента был следующий — человеку давалось по специальному резиновому мячику в каждую руку. От него требовалось сжимать эти мячики произвольно выбранной рукой в произвольные моменты времени, то есть без стимулов — тем самым авторы намеревались уменьшить влияние сигналов, поступающих от глаз или других органов чувств. Нарезка сигнала производилась с помощью показаний дополнительных датчиков электромиографии (EMG), которые фиксируют электрические сигналы непосредственно от сокращения мышц.

- Berlin BCI II-IV [29]

  Датасет также имеет структуру попыток и является как самым маленьким, так и самым старым среди представленных. Подопытные садились в обычный стул и клали руки в удобную позицию на клавиатуру. Их задачей было нажимать клавиши указательным пальцем и мизинцем в произвольном порядке и темпе. Эксперимент состоял из 3 сессий по 6 минут каждая с небольшими перерывами между ними (порядка нескольких минут). Данные, как и в случае с датасетом Ewan Nurse, были нарезаны с помощью сигнала EMG.

6 Эксперимент

Целью эксперимента был выбор локально оптимальной модели с помощью кросс-валидации и проверка гипотез 4.1, 4.2, 4.3.

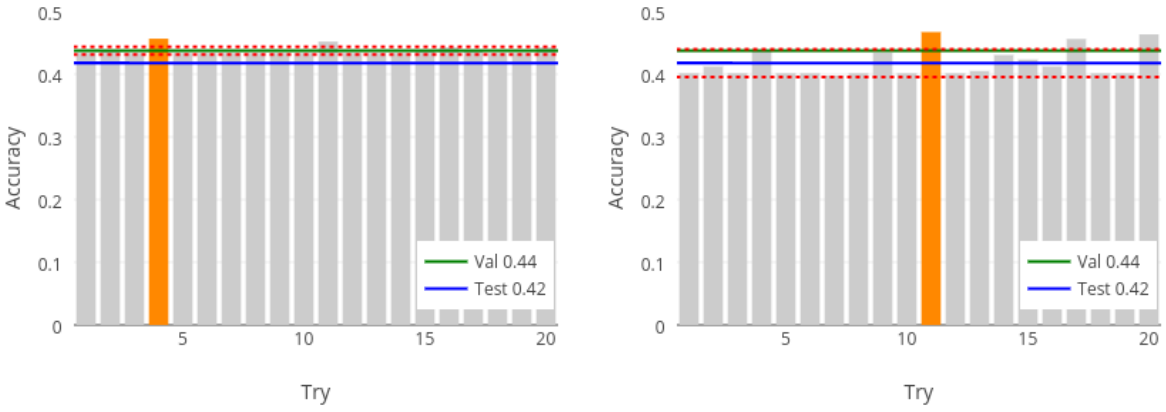
6.1 Обзор оригинальных решений

Отметим, что к каждому из датасетов прилагается авторское решение либо же решения, победившие в конкурсах. В случае двух датасетов Berlin BCI это были традиционные методы, поэтому в этой работе они не рассматриваются.

Решение [28], продемонстрированное авторами набора данных Ewan Nurse, использует обычный многослойный перцептрон. Для отбора лучшей структуры сети, т.е. нахождения $\hat{\mathbf{h}}$, авторы применяют генетический алгоритм. Число слоев сети ограничено 3, а число нейронов в слое — 100. Таким образом, описание одной сети состоит из трех чисел и пространство $\mathcal{H} = [0; 100]^3$. Затем случайным образом генерируются

30 описаний сетей и помещаются в нулевое поколение. На каждом шаге генетического алгоритма, чтобы отобрать лучшее особи для создания потомства, необходимо обучить все сети в поколении. Обучение производится с помощью обычного метода обратного распространения ошибки (BackPropagation) с модификацией алгоритмом имитации отжига (Simulated Annealing) [30] — после обучения сети ко всем весам добавляется случайные составляющие и сеть обучается еще, но уже начиная с этого приближения весов. Такой цикл повторяется 10 раз. Это позволяет снизить вероятность застопоривания в локальном минимуме.

В настоящей работе автор попробовал воспроизвести предложенный алгоритм с генетическим способом отбора моделей, а также с помощью алгоритма Tree Parzen Estimator [31]. Недостаток генетического метода в том, что приходится обучать очень много сетей, а учитывая, что используется связка BP+SA, это занимает очень много времени. Этот минус устраняется вторым алгоритмом. Но главный недостаток, который принципиально не могут устранить алгоритмы выбора модели — нестабильность обучения самой модели, т.е. очень высокая ошибка оценивания и низкая точность нахождения $f^*_{\mathcal{F}}$.



(a) Валидационная выборка

(b) Тестовая выборка

Рис. 4: Качество работы сети Ewan Nurse

Чтобы продемонстрировать это, проведем следующий эксперимент — будем обучать одну и ту же сеть несколько раз и смотреть на итоговое качество, которое получается. Изначально отсечем случайным сбалансированным по классам образом 20%

данных на тест. Алгоритм имитации отжига для своих внутренних шагов требует валидационной выборки, ее отсечем таким же образом в размере 20% от оставшихся данных. Обучая сеть с помощью BP+SA будем получать некоторое качество на тестовой и на валидационной выборке, по последнему из которых и надо отбирать наилучшую модель. Для достоверности будем обучать одну и ту же сеть 20 раз. Полученные результаты для одной из архитектур можно увидеть на рисунках 4a, 4b. Как видно из этих рисунков, различия в валидационном качестве конкретно для этой сети не велики, но в других экспериментах составляли до 5–7%. Более существенно здесь то, что качество на тестовой выборке скачет очень сильно и в итоге, выбирая лучшую модель по валидации, мы очень часто будем ошибаться и иногда эта ошибка будет существенной — в районе 5%. В этом случае любой выбор модели оказывается довольно бессмысленным. Поэтому данную модель также стоит отбросить.

Остается единственная модель, которая изначально и предполагалось исследовать в деталях — с соревнование Kaggle EEG.

6.2 Выбор оптимальной модели

Рассмотрим в подробностях модель, архитектура которой в общих чертах похожа на решение участников, занявших 3 место в соревновании Kaggle EEG [32].

Полную структуру сети можно увидеть на рисунке 5. Опишем по порядку, чем отличаются слои и зачем нужный каждый из них.

- На вход сети подается сигнал почти в исходной форме. Он отличается от сигнала с аппаратуры только применением полосового фильтра для отсечения той части спектра, в которой происходит основная мозговая деятельность. Первый сверточный слой осуществляет понижение размерности в канальной области. То есть, если изначально был c-канальный сигнал из n точек по времени, то первая свертка будет производиться именно по каналам и на выходе слоя будет меньшее число каналов, конкретно в этой модели 6. Как уже упоминалось в разделе 4.1, показания с различных но близких каналов сильно коррелированы, поэтому осуществление понижения размерности разумный шаг, который осуществляется и в традиционной схеме, но здесь реализуется с помощью одного сверточного слоя. Также этот этап дает возможность обучиться фильтрам таким образом, чтобы снизить зашумленность данных.

- После понижения размерности в канальной области, производится свертка по времени с целью также понизить размерность и агрегировать информацию в меньшем числе точек. По сути своей, это даунсемплинг по времени, т.е. понижение частоты временного ряда, но более сложным способом, чем просто выкидывание точек.
- Далее следует стандартный стекинг из сверточных и пулинг слоев для получения нового представления данных. Цель такого участка сети — выделить сложные паттерны из сигнала. Также отметим, что на схеме не изображен промежуточный этап с расширением признаковой размерности до 64 и последующего сжатия обратно до 32 путем пулинга по канальному измерению. Это сделано для того, чтобы еще больше усилить акцент именно на наличии какого-то паттерна, а не на его точном положении и, также, для того, чтобы снизить переобучение.
- Последний участок сети — полносвязные классификационные слои. Между ними стоят слои Dropout для снижения вероятности переобучения, ведь имено здесь сосредоточено наибольшее число параметров. Отметим, что в качестве активационной функции выбирается ReLU, т.к. помимо других плюсов, обучение с ней происходит

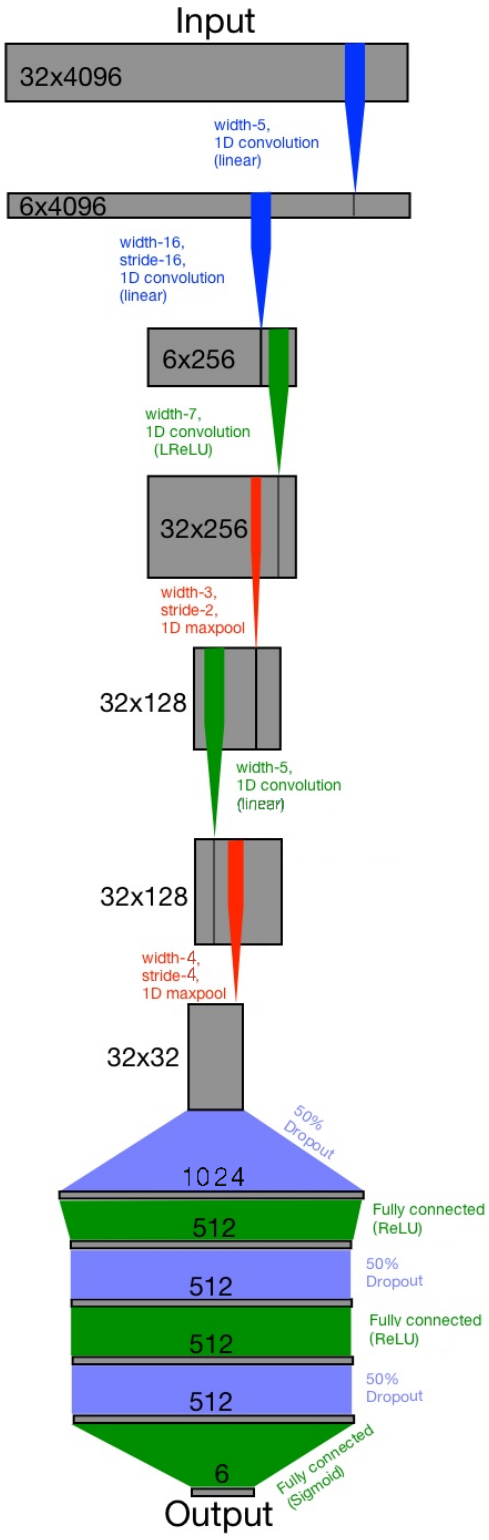
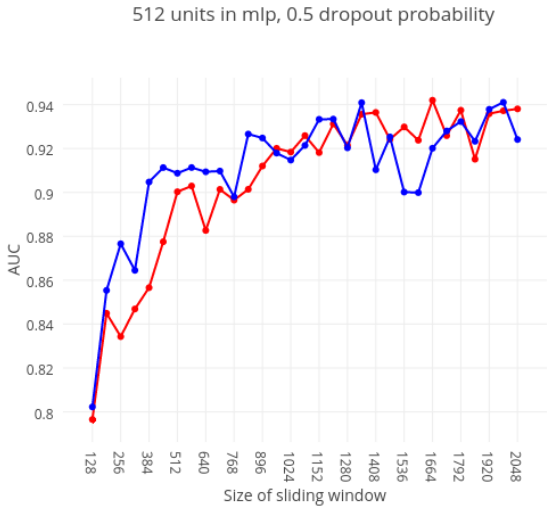


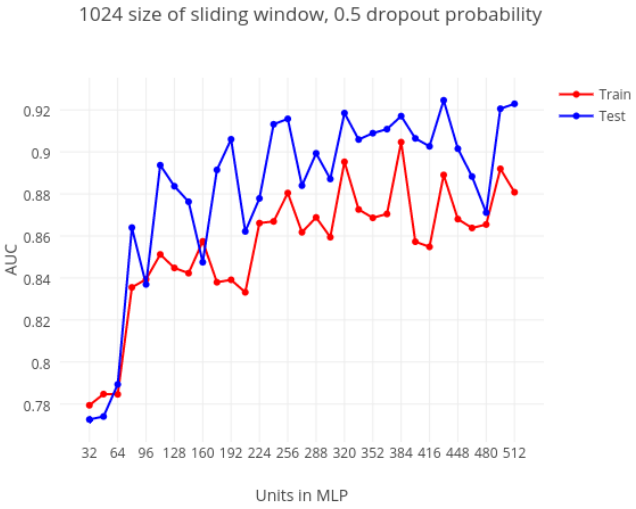
Рис. 5: Архитектура исходной сети

заметно быстрее, ибо она очень легко вычислима [11]. В последнем слое используется сигмоида, т.к. необходима вероятностная интерпретация результата.

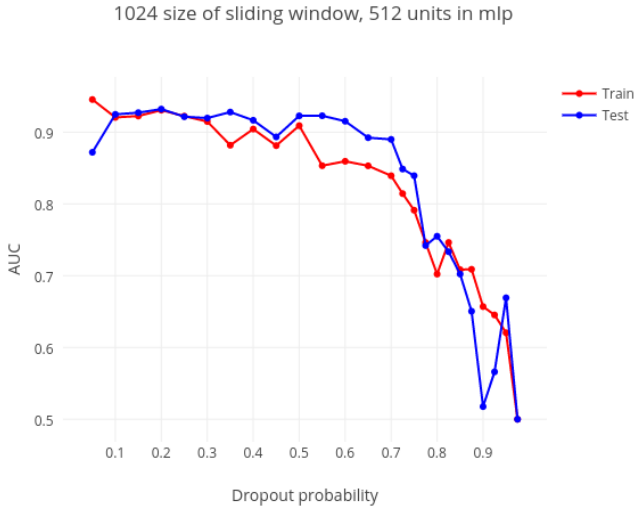
Будем искать локально оптимальную архитектуру в окрестности описанной только что модели. Выбирать будем из моделей, которые получаются из начальной изменением описанных в разделе 4.2 параметров. Способ выбора наилучшей архитектуры $\hat{\mathbf{h}}$ — кросс-валидация (см. раздел 2.2.2), метод построения лучшей модели $f^*_{\mathcal{F}_{\mathbf{h}}}$ в выбранном классе $\mathbf{h}$ — метод обратного распространения ошибки. Полученные результаты можно увидеть на рисунках 6, 7



(a) Размер окна

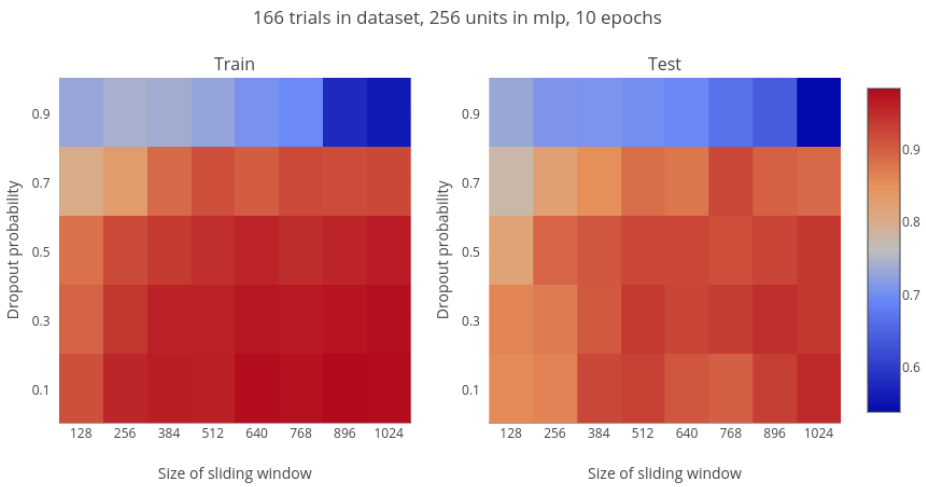


(b) Нейронов в полносвязных слоях

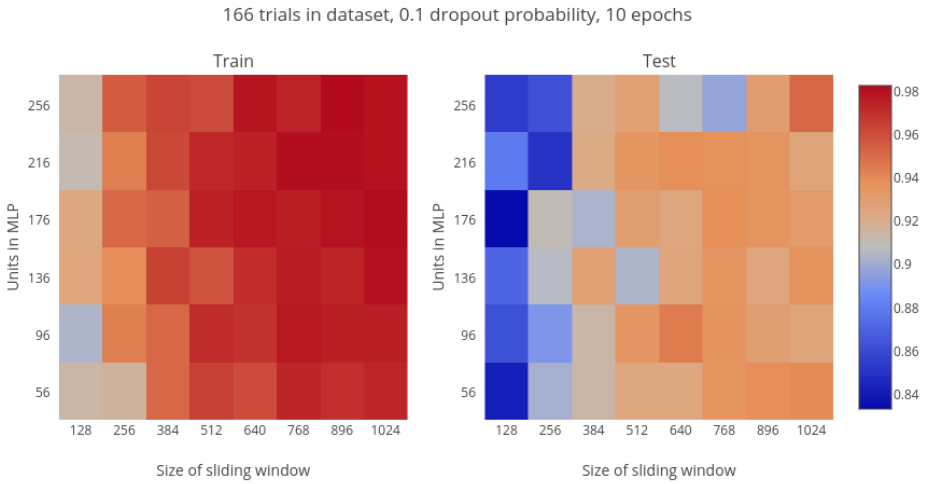


(c) Вероятность Dropout

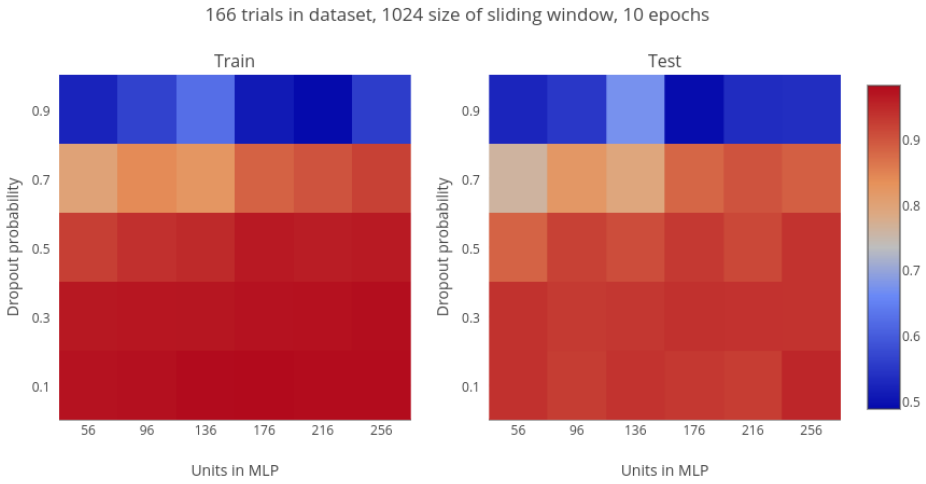
Рис. 6: Зависимость качества AUC от параметров в 1D–представлении



(a) Размер окна/Dropout



(b) Размер окна/Число нейронов



(c) Число нейронов/Dropout

Рис. 7: Зависимость качества AUC от параметров в 2D–представлении

Данные графики показывают лишь срезы трехмерного распределения качества по параметрам. По результатам анализа всех совместных вариантов параметров была выбрана наилучшая модель со следующими параметрами: размер окна = 2048, число нейронов в полносвязных слоях = 432, вероятность dropout = 0.35.

Но главное в этих графиках отметить то, что качество сначала примерно постоянно, а затем начинает монотонно падать с упрощением модели. Это значит, что почти без потери качества можно упростить исходную модель. А при одном и том же качестве в приоритете всегда более простая модель.

6.3 Перенос типа модели

7 Заключение

Список литературы

[1] *Mohri M., Rostamizadeh A., Talwalkar A.* Foundations of Machine Learning. — The MIT Press, 2012. — 432 pp.

[2] *Vapnik V. N.* Statistical Learning Theory. — Wiley-Interscience, 1998. — 768 pp.

[3] *Lugosi G., Wegkamp M.* Complexity regularization via localized random penalties // *The Annals of Statistics*. — 2004. — Vol. 32, no. 4.

[4] *Koltchinskii V.* Rademacher penalties and structural risk minimization // *IEEE Transactions on Information Theory*. — 2001. — Vol. 47, no. 5.

[5] *Hastie T., Tibshirani R., Friedman J.* The Elements of Statistical Learning: Data Mining, Inference, and Prediction. — Springer, 2009. — 745 pp.

[6] *Bartlett P. L., Maiorov V., Meir R.* Almost linear vc dimension bounds for piecewise polynomial networks // Advances in Neural Information Processing Systems 11. — The MIT Press, 1999. — Pp. 190–196.

[7] *Bartlett P. L., Maass W.* Vapnik-chervonenkis dimension of neural nets // The Handbook of Brain Theory and Neural Networks, 2nd ed. — The MIT Press, 2003. — Pp. 1188–1192.

[8] *Anthony M., Bartlett P. L.* Neural Network Learning: Theoretical Foundations. — Cambridge University Press, 1999. — 404 pp.

[9] *Goodfellow I., Bengio Y., Courville A.* Deep Learning. — Book in preparation for MIT Press.

[10] *Karpathy A.* Cs231n convolutional neural networks for visual recognition. http://cs231n.github.io/convolutional-networks/.

[11] *Glorot X., Bordes A., Bengio Y.* Deep sparse rectifier neural networks // Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011). — 2011. — Pp. 315–323.

[12] *He K., al.* Delving deep into rectifiers: Surpassing human-level performance on imagenet classification // 2015 IEEE International Conference on Computer Vision (ICCV). — 2015. — Pp. 1026–1034.

[13] Keras layers. http://keras.io/layers/core/.

[14] *Krizhevsky A., Sutskever I., Hinton G. E.* Imagenet classification with deep convolutional neural networks // Advances in Neural Information Processing Systems 25. — 2012. — Pp. 1097–1105.

[15] *Simonyan K., Zisserman A.* Very deep convolutional networks for large-scale image recognition // *CoRR*. — 2014. — Vol. abs/1409.1556.

[16] *He K., al.* Deep residual learning for image recognition // *CoRR*. — 2015. — Vol. abs/1512.03385.

[17] *Nicolas-Alonso L. F., Gomez-Gil J.* Brain computer interfaces, a review // *Sensors*. — 2012. — Vol. 12, no. 2. — Pp. 1211–1279.

[18] *Tangermann M., al.* Review of the bci competition iv // *Frontiers in neuroscience*. — 2012.

[19] *Fatourechi M., al.* Emg and eog artifacts in brain computer interface systems: A survey // *Clinical Neurophysiology*. — 2007. — Vol. 118, no. 3. — Pp. 480–494.

[20] *Hyvarinen A., Oja E.* Independent component analysis: algorithms and applications // *Neural Networks*. — 2000. — Vol. 13, no. 4–5. — Pp. 411–430.

[21] *Ang K. K., al.* Filter bank common spatial pattern (fbcsp) in brain-computer interface // 2008 IEEE International Joint Conference on Neural Networks. — 2008. — Pp. 2390–2397.

[22] *Arvaneh M., al.* Optimizing spatial filters by minimizing within-class dissimilarities in electroencephalogram-based brain–computer interface // *IEEE Transactions on Neural Networks and Learning Systems*. — 2013. — Vol. 24, no. 4. — Pp. 610–619.

[23] *Srivastava N., Hinton G., al.* Dropout: A simple way to prevent neural networks from overfitting // *Journal of Machine Learning Research*. — 2014. — Vol. 15. — Pp. 1929–1958.

[24] *Luciw M. D., Jarocka E., Edin B. B.* Multi-channel eeg recordings during 3,936 grasp and lift trials with varying weight and friction // *Scientific Data*. — 2014. — no. 140047.

[25] *Cukierski W.* Grasp and lift trial. https://www.youtube.com/watch?v=y3_Izuop2gY.

[26] *Blankertz B., al.* The non-invasive berlin brain–computer interface: Fast acquisition of effective performance in untrained subjects // *NeuroImage*. — 2007. — Vol. 37, no. 2. — Pp. 539 – 550.

[27] *Li Y. N., Zhang X. D., Huang Z. X.* A practical method for motor imagery based real-time prosthesis control // 2012 IEEE International Conference on Automation Science and Engineering (CASE). — 2012. — Pp. 1052–1056.

[28] *Nurse E. S., Karoly P. J., al.* A generalizable brain-computer interface (bci) using machine learning for feature discovery // *PLoS ONE*. — 2015. — Vol. 10, no. 6.

[29] *Blankertz B., Curio G., Muller K.-R.* Classifying single trial eeg: Towards brain computer interfacing // Advances in Neural Information Processing Systems 14. — MIT Press, 2002. — Pp. 157–164.

[30] *Schneider J., Kirkpatrick S.* Stochastic Optimization. — Springer, 2006. — 568 pp.

[31] *Bergstra J. S., Bengio Y., al.* Algorithms for hyper-parameter optimization // Advances in Neural Information Processing Systems 24. — 2011. — Pp. 2546–2554.

[32] Kaggle grasp-and-lift eeg detection. https://www.kaggle.com/c/grasp-and-lift-eeg-detection.