**Тезисы**

**«Эксперименты с алгоритмами регуляризации, нормализации, максимального подобия нейронных сетей»**

*Авторы*:

Склянов Семён Игоревич,

Россия, Иркутская область, г. Черемхово,

МОУ Лицей г. Черемхово, 11 класс

*Руководители*:

Катаев Виктор Борисович

учитель информатики высшей квалификационной категории

МОУ Лицей г. Черемхово,

Зотов Игорь Николаевич,

доцент кафедры самолётостроения и эксплуатации авиационной техники ФГБОУ ВО «Иркутский национальный исследовательский технический университет»,

кандидат технических наук.

г. Усолье-Сибирское, Иркутская область

2023 г.

**«Эксперименты с алгоритмами регуляризации, нормализации, максимального подобия нейронных сетей»**

Склянов Семён Игоревич*,*

*Россия, Иркутская область, г. Черемхово, МОУ Лицей г. Черемхово, 11 класс*

«Человечество способно уже сейчас

создать самую большую нейросеть.

К середине этого столетия нейросеть

по числу нейронов достигнет размеров человека»

*Александр Сербул, 1С-Битрикс*

**Введение**

**Актуальность** разработки данного проекта определена следующими факторами. Рост интереса к нейронным сетям: в последние годы нейронные сети стали ключевой технологией в многих областях, таких как машинное обучение, компьютерное зрение, обработка естественного языка и биоинформатика. Понимание и оптимизация работы нейронных сетей стали критически важными.

1. Проблема переобучения: Одной из основных проблем, с которой сталкиваются исследователи и разработчики, является переобучение (overfitting). Алгоритмы регуляризации могут помочь бороться с этой проблемой и повысить обобщающую способность моделей.
2. Стабильность обучения: Нормализация и методы максимального подобия могут улучшить стабильность обучения нейронных сетей, что делает их более предсказуемыми и легче настраиваемыми.
3. Повышение производительности: Оптимизация производительности нейронных сетей имеет большое значение для решения сложных задач, таких как обработка изображений, обработка текста и автоматическое управление. Эксперименты с алгоритмами могут помочь выявить наилучшие методы для ускорения обучения и вывода.

Проблема, которая вырисовывалась из первых примеров обучения нейронной сети давала почву для размышления и исследований.

В своей лекции №7 на кафедре математического обеспечения и применения ЭВМ в СПбГЭТУ "ЛЭТИ" авторы описывают, что главная проблема машинного обучения состоит в том, что алгоритм должен хорошо работать на новых данных, которых он раньше не видел, а не только на тех, что использовались для обучения модели [6]. Эта способность правильной работы на ранее не предъявлявшихся данных называется обобщением. И я с ними полностью согласен.

***Проблема.***Не всякий алгоритм обучения нейронной сети совершенен. Нужна тонкая настройка обучения или переобучения нейронных сетей и их моделей, связанная с модернизацией самой модели и её слоёв.

***Цель.*** Поэтому возникла цель исследовать алгоритмы обучения нейронных сетей, внести в алгоритмы свои параметры обучения. Изучить универсальность методов и путем эксперимента найти универсальные шаги обучения нейросетей.

Для этого я поставил перед собой следующие задачи:

1. Изучить основные методы обучения нейросетей.
2. Анализировать способность этих сетей к переобучению по новой модели.
3. Найти параметры методов, способные воздействовать на построение модели.
4. Провести эксперименты смещение слоёв модели и воздействия этих параметров для достижения более точных результатов
5. Найти оптимальный алгоритм обучения нейросети.

*Новизна* проекта заключается в том, что чаще всего алгоритмы, представленные для обучения нейронной сети, преследуют одну цель, а построенные модели не могут быть универсальным средством обучения. Мои эксперименты могут помочь построению новых моделей обучения.

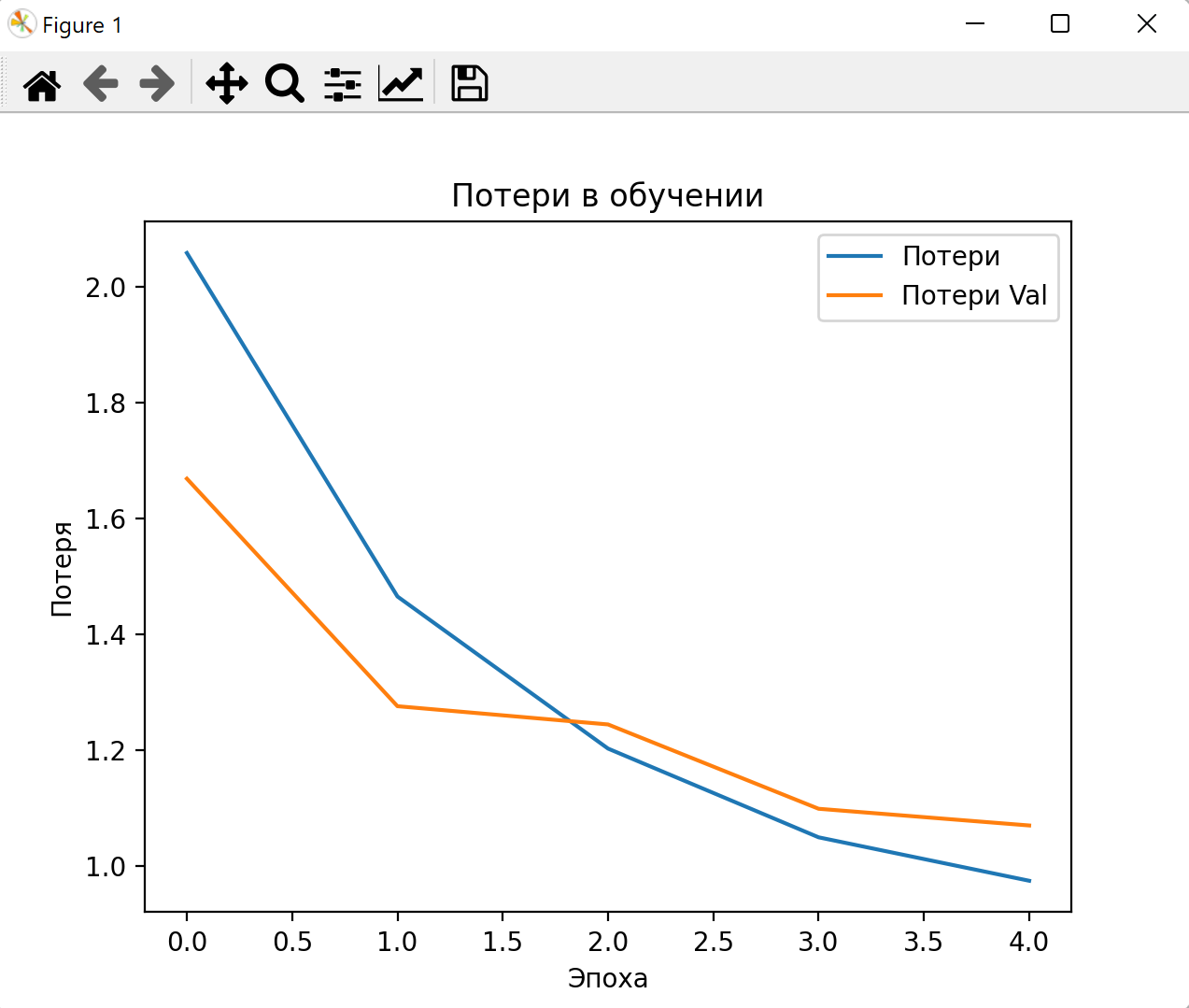
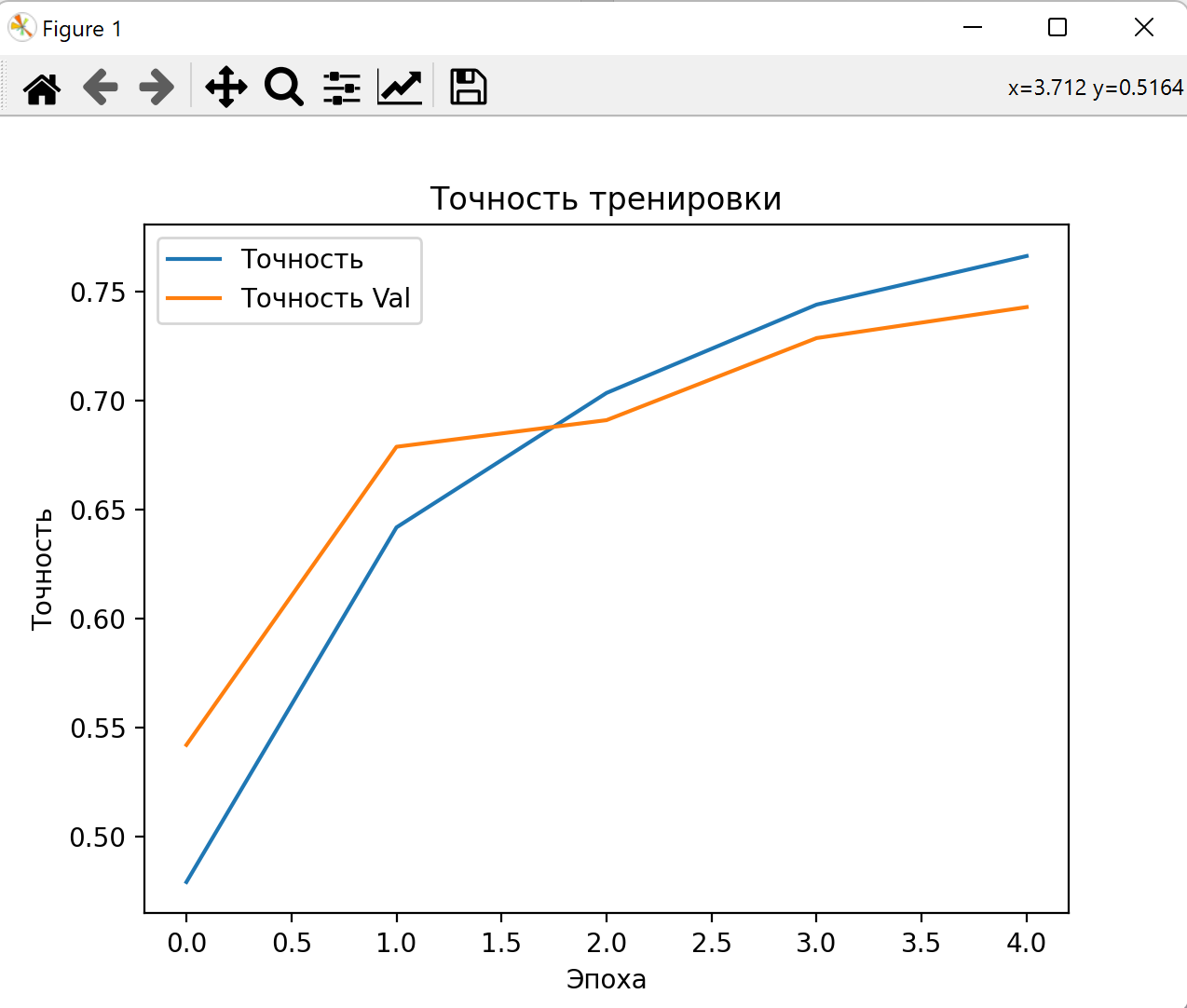
В результате экспериментов создания моделей и обучения нейронных сетей мы наблюдали эффективные алгоритмы резкого снижения потерь в обучении, и повышение точности, но также можем наблюдать значительное замедление процесса обучения. Около 110 секунд в среднем требуется для обучения в один проход, т.е. в одну эпоху. Вот здесь возникает вопрос применения аппаратных вычислений, потому что один центральный процессор не справится с такой сложной задачей. Поэтому для L-2 регуляризации рекомендуют использовать графические видеокарты, способные в значительной степени увеличить скорость обучения. И это еще одна из интересных тем, которая достойна внимания. В пределах этого проекта мы не будем рассматривать процесс ускорения, ведь мы преследуем иные цели- найти алгоритм и совокупность, комбинацию применения методов обучения нейронных сетей.

**Результат проекта.**

**Алгоритм построения модели по комбинации удачных приёмов и методов обучения.**

Здесь, внутри программы, в области комментариев я постарался более полно описать назначение каждого слоя модели и эффект применения в обучении. import tensorflow as tf  
from tensorflow import keras  
from tensorflow.keras import layers  
from tensorflow.keras.callbacks import EarlyStopping  
import matplotlib.pyplot as plt  
***# Загружаем 70 000 изображений 60 000 тестовых и 10 000 тренировочных***(x\_train, y\_train), (x\_test, y\_test) = keras.datasets.mnist.load\_data()  
***# Нормализация данных (упрощение и сведение к одному диапазону от 0 до 1 и от -1 до 1 сводим к диапазону от 0 до 1) Всё что меньше 1 становится 0, всё что больше 1 становится 1***x\_test.astype("float32") / 255.0  
x\_train.astype("float32") / 255.0  
  
***# Создаём структуру модели***model = keras.Sequential([  
 layers.Flatten(input\_shape=(28, 28)),  
***# Переводим из многомерного массива в одномерный c 28 мерным вектором  
 # Это первый слой нашей нейронной сети. Каждое изображение имеет 28 \* 28 = 784 значения и такFlatten()  
 # а изображения представлены в виде массивов размером 28x28, заполненных значениями пикселей***layers.Dense(128, activation="relu"),  
 ***# Создаем свой слой с 28 нейронами c функцией активации "relu",  
 # Это другой вид слоя, который мы видим в модели, создан с использованием tf.keras.layers.Dense(). Он создает то, что называется полностью связанным или плотно связанным слоем. Это можно сравнить с разреженным слоем, и различие связано с тем, как информация передается между узлами в соседних слоях.  
 # Первый параметр (128 в первом случае) указывает, сколько узлов должно быть на уровне.  
 # Число узлов в скрытых слоях (слоях, которые не являются входными или выходными слоями) несколько произвольно, но важно отметить, что выходной слой имеет количество узлов, равное количеству классов, которые модель пытается предсказать.***

***Это очень важно, потому что на выходе для каждого узла  
 # конечного слоя будет вероятность того, что данное изображение является конкретной цифрой.***layers.Dense(128, activation="relu"),  
 ***# Уровень в нашей сети называется «ReLU», что означает сокращение выпрямленной линейной единицы.  
 # Что делает ReLU, так это делает активацию любых отрицательных логитов 0 (узел не срабатывает), при этом оставляя любые положительные логиты неизменными (узел стреляет с силой, линейно пропорциональной силе входа).***layers.Dropout(0.2),***# Dropout - это метод который случайным образом отключает какой-либо нейрон каждую итерацию, т.е каждую итерацию прохождения эпохи. Это помогает, исключить пререобучение. Dropout(0.5) - 50% нейронов отключится  
 # С вероятностью 20% будем отключать по 1 нейрону, чтобы избежать переобучения нейронной сети* *# Уменьшаем количество нейронов до 128***layers.Dense(128, activation="relu"),  
 layers.Dropout(0.2),  
 ***# Концепция отсева восходит к более раннему обсуждению связности слоев и имеет отношение конкретно к нескольким недостаткам, связанным с плотно связанными слоями. Одним из недостатков плотно связанных слоев является то, что это может привести к очень дорогостоящим в вычислительном отношении нейронным сетям.  
 # С каждым узлом, передающим информацию каждому другому узлу на следующем уровне, сложность взвешенных сумм, вычисленных в каждом узле, экспоненциально увеличивается с количеством узлов в каждом уровне.  
 # Еще один недостаток заключается в том, что при передаче большого количества информации от слоя к слою модели могут иметь тенденцию превышать данные обучения, что в конечном итоге снижает производительность.******# Таким образом, в записной книжке для начинающих, вызов Dropout(0.2) между двумяDense()слои делает так, что каждый узел в первом Dense()слое имеет вероятность 0,2 быть исключенным из расчета активаций следующего слоя.  
 # Возможно, вы поняли, что это эффективно делает выходной слой в модели слабо связанным слоем.******# Заканчиваем преобразованием в придикты***layers.Dense(10, activation="softmax"),  
  
])  
***# эрлистопинг - это когда один из нейронов не приносит пользы, тот нейрон, который пытается переобучиться:  
# Когда потери будут составлять 2, отключим нейрон***early = EarlyStopping(monitor="val\_loss", patience=2)  
  
***# Следующий этап - это Компиляция модели (оптимизация функции потерь (насколько нейронка была менее точной предыдущей модели)  
# Укажем конфигурацию обучения (оптимизатор, функция потерь, метрики)  
# True для уменьшения) Определим метрику точности "accuracy"  
# optimizer="Adam" - среднеквадратичное распределение  
# loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True) - Минимизируемая функция потерь***model.compile(optimizer="adam", loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),  
 metrics=["accuracy"])  
***# Функция вызывается по заранее построенной модели и определяет функцию потерь, оптимизатор и метрики, каждую из которых хочу объяснить. Это важные особенности того, как нейронная сеть дает свои окончательные прогнозы.  
# Функция потерь является частью модели, которая количественно определяет, насколько далеко от прогноза находится правильный ответ. Различные типы моделей будут нуждаться в различных функциях потерь.  
# Например, функция потерь для такой проблемы, как эта, где выходные данные для нашей модели являются вероятностями, должна сильно отличаться от функции потерь модели, которая пытается предсказать.  
# Функция потерь для этой конкретной модели - sparse\_categorical\_crossentropy, хороша для задач классификации мультикласса, подобных этой. В нашем случае, если модель предсказывает, что изображение имеет лишь небольшую вероятность быть его фактической меткой, это приведет к большим потерям.  
  
# Создадим объект взаимодействия истории  
# Получение модели (через цикл обучения модели - эпохи чем больше, тем лучше)  
# validation\_split=0.1 - тонкая настройка обучения, с функцией поиска переобученного нейрона  
# Мы передаем валидационные данные для x\_train, y\_train***history = model.fit(x\_train, y\_train, epochs=10, validation\_split=0.1, callbacks=[early])  
  
plt.plot(history.history["loss"], label="Потери")  
plt.plot(history.history["val\_loss"], label="Потери Val")  
plt.title("Потери в обучении")  
*# Подпишем оси*plt.xlabel("Эпоха")  
plt.ylabel("Потеря")  
plt.legend()  
plt.show()

 Графики

7а – «Потери в обучении 6б – «Точность обучения с

Моего алгоритма моего алгоритма

Epoch 1/5

704/704 [==============================] - 103s 144ms/step - loss: 2.0593 - accuracy: 0.4794 - val\_loss: 1.6691 - val\_accuracy: 0.5422

Epoch 2/5

704/704 [==============================] - 104s 147ms/step - loss: 1.4654 - accuracy: 0.6419 - val\_loss: 1.2757 - val\_accuracy: 0.6788

Epoch 3/5

704/704 [==============================] - 103s 147ms/step - loss: 1.2025 - accuracy: 0.7035 - val\_loss: 1.2442 - val\_accuracy: 0.6910

Epoch 4/5

704/704 [==============================] - 103s 147ms/step - loss: 1.0491 - accuracy: 0.7439 - val\_loss: 1.0984 - val\_accuracy: 0.7286

Epoch 5/5

704/704 [==============================] - 104s 148ms/step - loss: 0.9740 - accuracy: 0.7662 - val\_loss: 1.0696 - val\_accuracy: 0.7428

Отличный результат! Уверенно могу сказать, что потери в обучении при такой комбинации применения методов становятся с каждой эпохой всё ниже и ниже, а точность данных при этом возрастает почти экспоненциально.

Теперь мы можем проверить любой код на скорость с новой комбинацией методов согласно моему алгоритму. Нельзя забывать и об уменьшении скорости обучения из-за метода L2 регуляризации, требующего аппаратных возможностей, но тем не менее цель достигнута, алгоритм создан.

**Результат проектной деятельности**

К описанию своего исследовательского проекта прилагаю успешные программные коды экспериментов и как результат мой алгоритм и модель успешного обучения нейросетей.

**Значимость проекта**

О значимости проекта судить рано, еще много работы. Надо обязательно проработать скоростные способы аппаратного взаимодействия с алгоритмом, но уверенно могу сказать, что обучение нейросети по моим экспериментам и итоговому алгоритму может пригодиться для изучения особенностей построения слоёв моделей нейросетей.

**Заключение**

В принципе, всё уже сказано. Модель сформирована, алгоритм создан и функционирует, хотя не очень быстро, но эффективно, снижая потери и увеличивая точность обучения нейронных сетей. Цель достигнута.