Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра суперкомпьютеров и квантовой информатики

Практическое использование нейронных сетей для бинарной классификации изображений

ОТЧЕТ ПО ПРАКТИКУМУ

Студент дневного отделения бакалавариата

Лю Хайлинь

Преподаватель:

канд. физ.-мат. наук, Буряк Д.Ю.

Москва, 2024

I. ПОДСТАНОВКА ЗАДАЧИ

Бинарно классифицировать изображении, где есть, либо нет солнцезащитных очков, изображения по формату 30x32, grayscale, jpg, обучающая выборка состоит из 378 изображений, а тестовая 22140.

II. ОСНОВНЫЕ ЭЛЕМЕНТЫ

Самый распространенная проблема, возникающая при обучении глубоких нейронных сетей переобучение представляет собой переобучение, к этой проблемы обычно вызвано несколькими причинами.

- Малый набор данных, для обучения такого набора недостаточно, модель может изучить шум в данных вместо признаков.
- Высокая сложность модели, слишком сложная модель имеет многие параметры, что может привести к изучению конкретных признаков в обучающих данных вместо общих.
- Слишком большое количество итерации при обучений модель может переучить данные обучения и не сможет адаптироваться к новым данным.
- Небалансные данные также является проблемой,. если в некоторых классах выборок гораздо больше, чем в других, модель может быть смещена в сторону более распространенных классов.

Так как размер обучающего набора строго ограничен, чтобы избежать переобучения, попробуем реализовать те стратегии, как регуляризация, аргументация данных, dropout, и упрощение модулей. Кроме того, использование кросс-валидации позволяет лучше оценить способность модели к обобщению.

2.1. Аргументация данных

Основные методы аргументации изображений основаны на таких преобразованиях изображений, как вращение, переворачивание и обрезка. Большинство из этих методов работают непосредственно с самим изображением и их легко реализовать (см. Таб.1).

Однако существуют и недостатки. Во-первых, применение базовой обработки изображений имеет смысл только при условии, что существующие данные подчиняются распределению, близкому к фактическим данным. Во-вторых, эффект заливки влияет на некоторые основные методы обработки изображений, такие как перемещение и вращение. То есть после операции некоторые участки изображения выводятся за пределы и теряются. Поэтому для заполнения пропусков реализуем некоторые действии.[1]

Методы	Описание
Переворот	Переверните изображение по горизонтали, вертикали или по обоим направлениям.
Поворот	Поворот изображения под углом.
Коэффициент масштабирования	Увеличение или уменьшение размера изображения.
Инъекция шума	Добавьте шум в изображение.
Цветовое пространство	Изменение цветовых каналов изображения.
Контрастность	Изменение контрастности изображения.
Резкость	Изменение резкости изображения.
Перевод	Переместите изображение по горизонтали, вертикали или в обоих направлениях.
Обрезка	Обрезка части изображения.

Таб.1 Основные методы аргументации изображений и краткое описание.[1]

2.2. L2 регуляризация

Несмотря на многочисленные успехи нейронных сетей в приложениях, переобучение остается одной из основных проблем, которую необходимо решить сообществу исследователей машинного обучения. Для решения этой проблемы было предложено несколько методов. Широко используемый метод заключается в добавлении члена регуляризации к функции ошибок, чтобы минимизировать общую функцию ошибок, в следующей форме:

$$\boldsymbol{\mathcal{L}}_T(\boldsymbol{W}) = \boldsymbol{\mathcal{L}}_{\boldsymbol{D}}(\boldsymbol{W}) + \lambda \boldsymbol{\mathcal{L}}_{\boldsymbol{W}}(\boldsymbol{W}) \quad (1)$$

где $\boldsymbol{W}$ — параметры нейронной сети. $\boldsymbol{\mathcal{L}}_T(\boldsymbol{W})$ — функция общих потерь, а λ — параметр регуляризации, который контролирует компромисс между $\boldsymbol{\mathcal{L}}_D(\boldsymbol{W})$ и $\boldsymbol{\mathcal{L}}_{\boldsymbol{W}}(\boldsymbol{W})$. $\boldsymbol{\mathcal{L}}_{\boldsymbol{D}}(\boldsymbol{W})$ представляет собой функцию суммы квадратов ошибок между целевым выходом $y(i)$ и выходом сети $h(x(i); \boldsymbol{W})$ и может быть рассчитана по следующей формуле:

$$\mathcal{L}_D(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^{m} [y^{(i)} - h(x^{(i)}; \mathbf{W})]^2 \#(2)$$

В последнем члене (1) $\boldsymbol{\mathcal{L}_W(W)}$ представляет собой сумму квадратов весовых параметров, которые можно определить по формуле:

$$\mathcal{L}_W(\mathbf{W}) = \frac{1}{2}\mathbf{W}^T\mathbf{W} \quad (3)$$

Такой особый вид регуляризации называется регуляризацией L2 (или распадом веса).[2]

2.3. Dropout

Dropout предотвращает переобучение и обеспечивает способ приблизительного эффективного объединения экспоненциально большого количества различных архитектур нейронных сетей. Термин «Dropout» относится к выпадающим единицам (скрытым и видимым) в нейронной сети. Временно удалив ячейку из сети вместе со всеми ее входящими и исходящими соединениями, как показано на рис. 1. Выбор нейронов для удаления является случайным. В простейшем случае каждый нейрон сохраняется с фиксированной вероятностью p, независимой от других нейронов, причем p можно выбрать с помощью набора владельцы или просто установить 0.5, что кажется близким к оптимальному для различных сетей.[3]

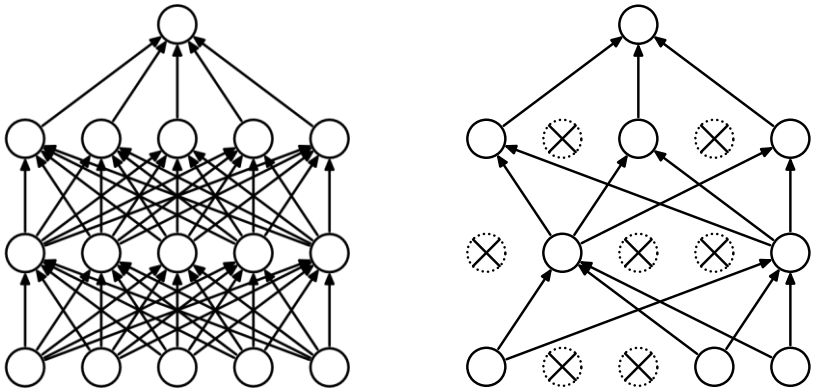

Рис.1 Модель нейронной сети с выпадением.[3]

Применение Dropout к нейронной сети равносильно выборке из нее прореженной сети, состоящей из всех единиц, переживших отсев. Нейронную сеть с n единицами можно рассматривать как набор из $2n$ возможных прореженных нейронных сетей. Все эти сети имеют одинаковые веса, так что общее количество параметров по-прежнему равно $O(n^2)$ или меньше. Для каждого представления каждого обучающего случая отбирается и обучается новая прореженная сеть. Таким образом, обучение нейронной сети с отсевом

можно рассматривать как обучение набора из $2n$ прореженных сетей с обширным распределением веса, где каждая прореженная сеть обучается очень редко, если вообще обучается.[3]

2.4. Кросс-валидация

Кросс-валидация — это метод, используемый в машинном обучении для оценки производительности модели на невидимых данных. Он включает в себя разделение доступных данных на несколько сгибов или подмножеств, использование одного из этих сгибов в качестве набора проверки и обучение модели на оставшихся свертках. Этот процесс повторяется несколько раз, каждый раз в качестве набора проверки используется другая складка. Наконец, результаты каждого этапа проверки усредняются, чтобы обеспечить более надежную оценку производительности модели. Перекрестная проверка — это важный шаг в процессе машинного обучения, помогающий гарантировать, что модель, выбранная для развертывания, является надежной и хорошо обобщается на новые данные.

Основная цель Кросс-валидации — предотвратить переобучение, которое происходит, когда модель хорошо обучена на обучающих данных, но плохо работает на новых, невидимых данных. Оценивая модель на нескольких наборах проверки, перекрестная проверка обеспечивает более реалистичную оценку эффективности обобщения модели, т. е. способности модели хорошо работать на новых, невидимых данных.[4]

III. ЭКСПЕРИМЕНТАЛЬНАЯ УСТАНОВКА

3.1. Программное и аппаратное обеспечение

Python[5] - бесплатный объектно-ориентированный язык программирования с открытым исходным кодом, привлекает внимание своим простым синтаксисом и динамической структурой. В Python принято создавать функции. Еще одним преимуществом является то, что он работает совместно со многими библиотеками, в которых можно создавать приложения «машинного обучения». В этой работе мы выбрали Python 3.11.8.

PyTorch[6] - платформа глубокого обучения с открытым исходным кодом, разработанная исследовательской группой Facebook по искусственному интеллекту. Он известен своей простотой использования, гибкостью и мощными функциями. PyTorch широко используется в областях искусственного интеллекта, таких как компьютерное зрение и обработка естественного языка, и имеет активное сообщество, которое предоставляет большое количество предварительно обученных моделей и инструментов для поддержки исследований и разработок.

CUDA[7] (Compute Unified Device Architecture) - это архитектура параллельных вычислений общего назначения, разработанная NVIDIA. Он позволяет разработчикам использовать графические процессоры NVIDIA для эффективных параллельных вычислений и особенно подходит для научных вычислений и задач глубокого обучения, которые обрабатывают большие объемы данных. CUDA позволяет выполнять сложные вычислительные задачи на графическом процессоре, предоставляя параллельную вычислительную среду, содержащую сотни ядер.

CuDNN[8] (библиотека глубоких нейронных сетей CUDA) - это библиотека ускорения графического процессора, специально разработанная NVIDIA для глубокого обучения. Он предоставляет ряд оптимизированных вычислительных функций нейронных сетей, таких как свертка, объединение, нормализация и уровни активации, для эффективного выполнения обучения и вывода моделей глубокого обучения на графических процессорах NVIDIA. cuDNN широко используется в различных средах глубокого обучения, таких как TensorFlow, PyTorch и т. д., для повышения производительности на графическом процессоре.

Критерием оценки эффективности нейронных сетей является время выполнения, которое зависит от производительности используемого компьютера. В связи с этим технические характеристики компьютера, использованного в настоящей работе, следующие:

Центральный процессор: 12-е поколение Intel(R) Core(TM) i7-12700H 2.30 ГГц

Оперативная память: 16 ГБ (используемая 15.7 ГБ)

Операционная система: Ubuntu 22.04.4 LTS

Графический процессор: NVIDIA GeForce RTX 3060 Laptop GPU

3.2. Настройка структуры нейронных сетей

Мы используем многослойную глубокую нейронную сеть, состоящую из двух сверточных слоев и двух слоев пула, за которыми следует выравнивание данных, а затем два полностью связанных слоя, чтобы построить два разных перцептрона на основе разных параметров.

В качестве функции потери выбираем `CrossEntropyLoss()`, а оптимизатор `Adam()`.

В `ConvNN_1` мы используем 8 ядер свертки 3*3 для свертки на первом слое свертки и выполняем максимальное объединение 3*3, а также используем 16 ядер свертки 3*3 для свертки на втором слое свертки.Произведение и выполняем максимальное объединение 2*2. сглаживая тензор, соединяем полносвязный слой с 40 нейронами, а выходной слой содержит 2 нейрона.

```
ConvNN_1(
  (conv1): Conv2d(1, 8, kernel_size=(3, 3), padding=(2, 1))
  (maxpool1): MaxPool2d(kernel_size=2)
  (activation): relu()
  (conv2): Conv2d(8, 16, kernel_size=(3, 3))
  (maxpool2): MaxPool2d(kernel_size=2)
  (activation): relu()
  (flatten): Flatten()
  (fc1): Linear(in_features=784, out_features=40)
  (activation): relu()
  (fc2): Linear(in_features=40, out_features=2)
  (activation): sigmoid()
)
```

В `ConvNN_2` мы используем 8 ядер свертки 3*3 для свертки на первом слое свертки и выполняем максимальное объединение 2*2, а также используем 16 ядер свертки 3*3 для свертки на втором слое свертки.Произведение и выполняем максимальное объединение 2*2. сглаживая тензор, соединяем полносвязный слой с 32 нейронами, а выходной слой содержит 2 нейрона.

```
ConvNN_2(
  (conv1): Conv2d(1, 8, kernel_size=(3, 3), padding=(1, 0))
  (maxpool1): MaxPool2d(kernel_size=3)
  (activation): relu()
  (conv2): Conv2d(8, 16, kernel_size=(3, 3))
  (maxpool2): MaxPool2d(kernel_size=2)
  (activation): relu()
  (flatten): Flatten()
  (fc1): Linear(in_features=256, out_features=32)
```

```
  (activation): relu()
  (fc2): Linear(in_features=32, out_features=2)
  (activation): sigmoid()
)
```

3.3. Оценка

В качестве метрик качества моделей используются следующие показатели:

True Positive (TP) – количество (процент) правильно классифицированного нормального трафика.

$$TP\ rate = \frac{TP}{TP + FN} \tag{4}$$

True Negative (TN) - количество (процент) правильно классифицированного аномального трафика.

$$TN\ rate = \frac{TN}{FP + TN} \tag{5}$$

False Negative (FN) - количество (процент) неправильно классифицированного нормального трафика.

$$FN\ rate = \frac{FN}{TP + FN} \tag{6}$$

False Positive (FP) - количество (процент) неправильно классифицированного аномального трафика.

$$FP\ rate = \frac{FP}{FP + TN} \tag{7}$$

Accuracy (доля верных ответов)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

F1 - это гармоническое среднее Precision и Recall. В то время как обычное среднее учитывает все значения одинаково, гармоническое среднее придает гораздо больший вес низким значениям

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{9}$$

IV. РЕЗУЛЬТАТ И ОБСУЖДЕНИЕ

4.1. Рассмотрение результатов обучении

Мы делим обучающую выборку и валидационную выборку в соотношении 80% и 20% и обучаем две нейронные сети[1] отдельно. Batch_size обучающего набора имеет значение 1, а Batch_size проверочного набора - 15. Преобразуем исходные данные изображения в тензор, регуляруем его до интервала [0, 1] и выполним итерацию 40 раз, скорость обучения 0.002. Записываем потери и точность прогнозирования после каждой итерации и рисуем кривую, чтобы интуитивно отражать процесс обучения.

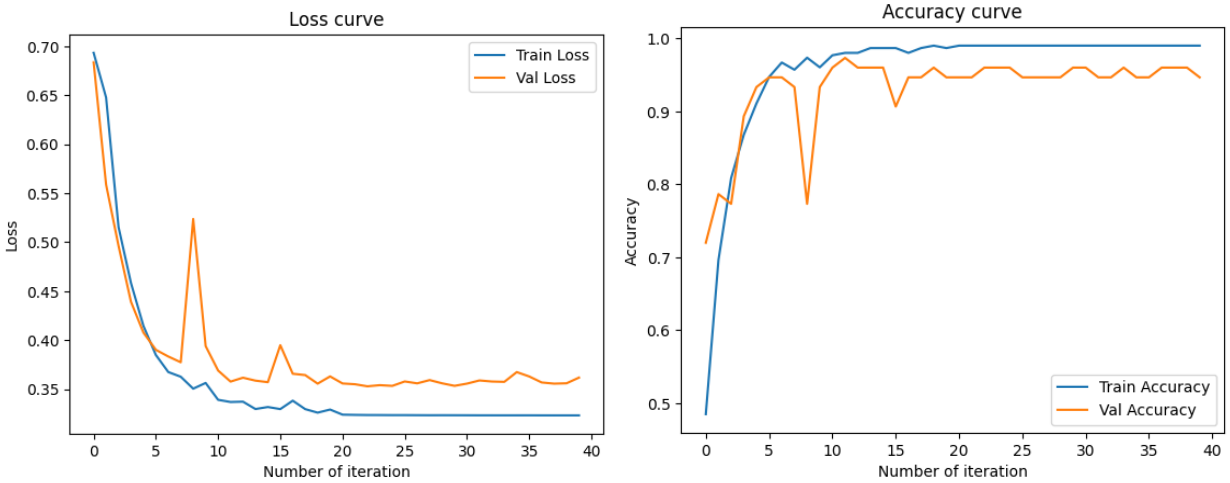

Рис.2 Loss и Accuracy модели ConvNN_1

Training loss: 0.313 Validation loss: 0.338 Training accuracy: 1.000 Validation accuracy: 0.976

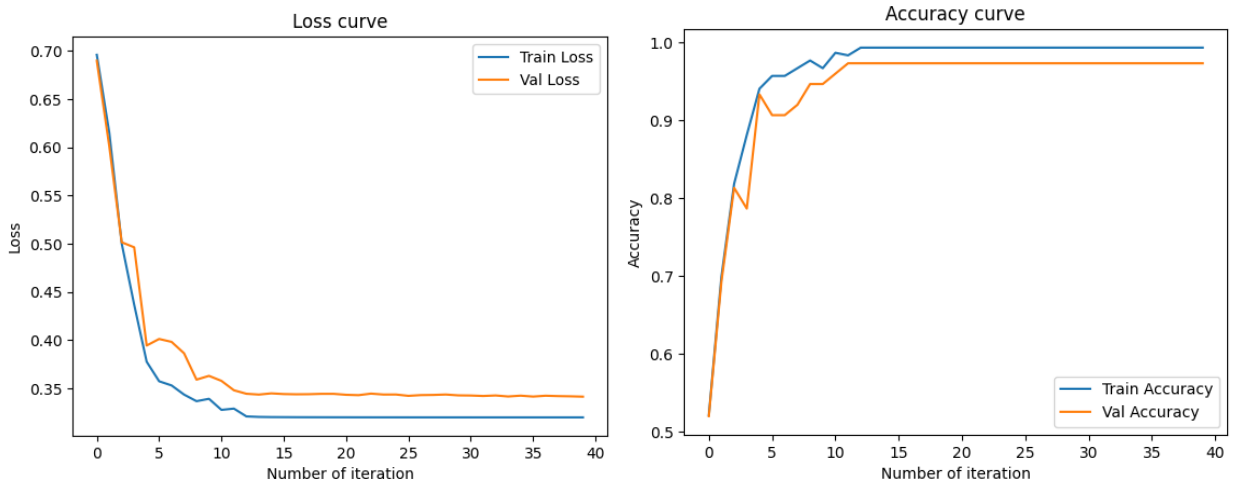

Рис.3 Loss и Accuracy модели ConvNN_2

Training loss: 0.320 Validation loss: 0.342 Training accuracy: 0.993 Validation accuracy: 0.973

Видно, что существует расхождение между обучающим набором и валидационным набором в обеих моделях, то есть потери на обучающем наборе уменьшаются, а потери на

[1] https://github.com/lhl4971/tenser_2024

проверочном наборе – нет, или точность на обучающем наборе увеличивается, но точность на проверочном не меняется – это проявление переобучения.

4.2. Реализация аргументации данных

Мы добавляем в набор данных 25% горизонтально перевернутых изображений и пытаемся снова обучить модель.

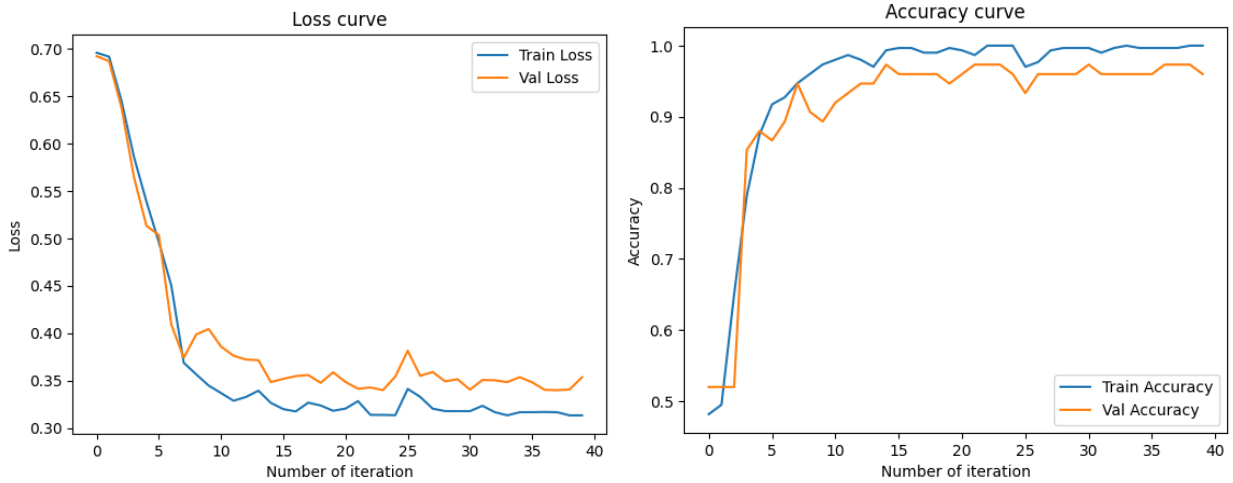

Рис.4 Loss и Accuracy модели ConvNN_1(Аргументация данных)

Training loss: 0.315 Validation loss: 0.344 Training accuracy: 0.998 Validation accuracy: 0.968

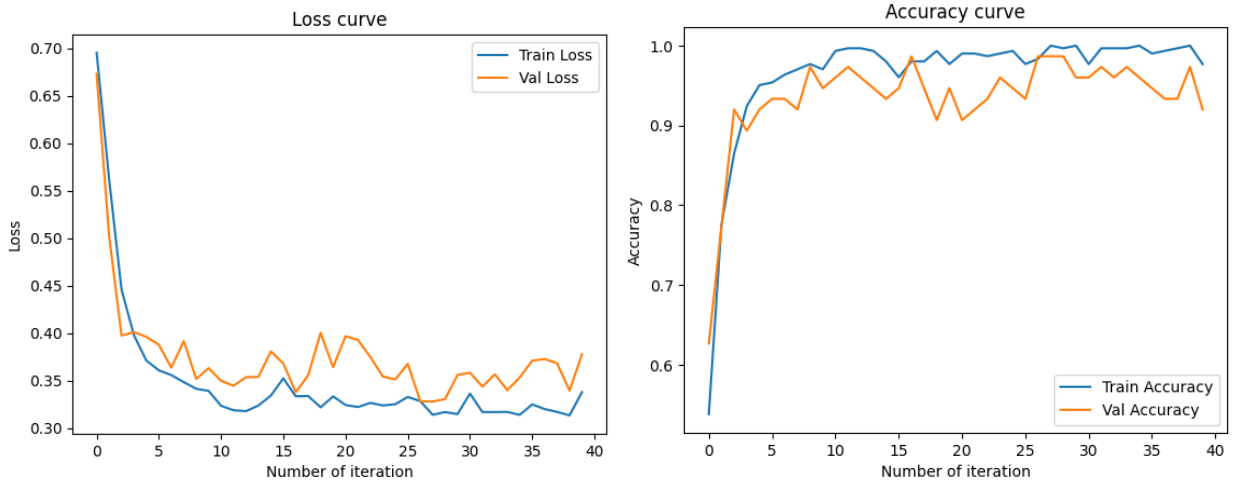

Рис.5 Loss и Accuracy модели ConvNN_2(Аргументация данных)

Training loss: 0.323 Validation loss: 0.366 Training accuracy: 0.991 Validation accuracy: 0.941

Мы обнаружили, что методы аргументации данных мало помогли при обучении на этом наборе данных.

4.3. Реализация L2 регуляризации

Из-за разницы в сложности моделей мы использовали разные параметры регуляризации для двух моделей, где `ConvNN_1` равен $1e-4$, а `ConvNN_2` равен $5e-6$.

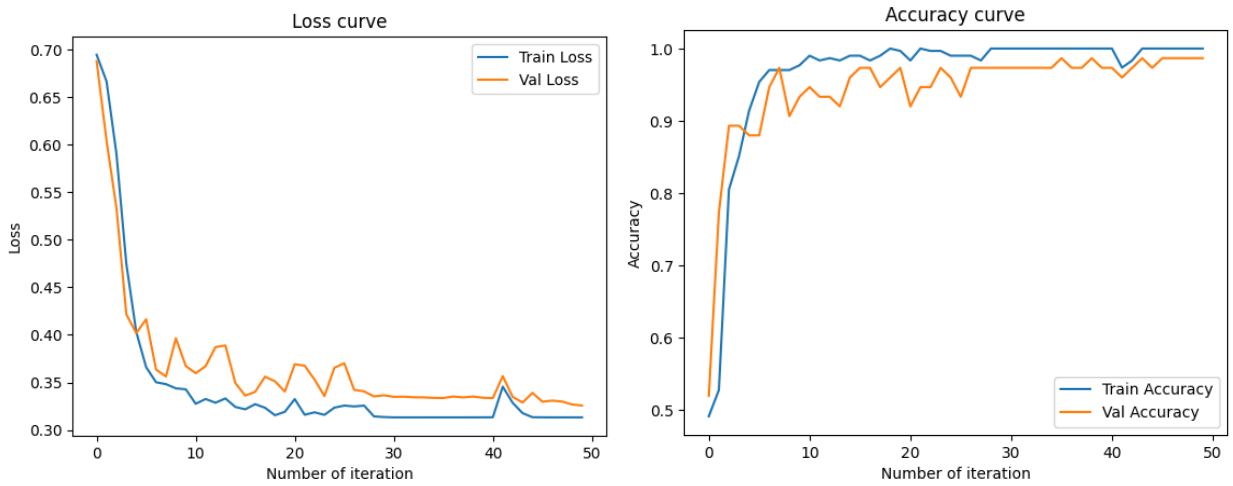

Рис.6 Loss и Accuracy моделя ConvNN_1 (L2 регуляризация)

Training loss: 0.313 Validation loss: 0.329 Training accuracy: 1.000 Validation accuracy: 0.987

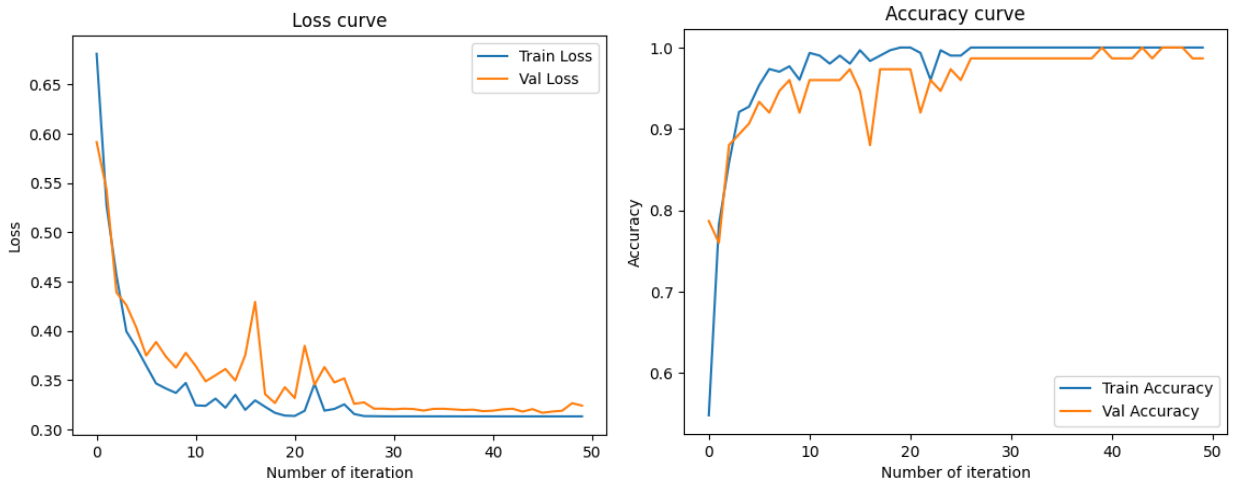

Рис.7 Loss и Accuracy моделя ConvNN_2 (L2 регуляризация)

Training loss: 0.313 Validation loss: 0.321 Training accuracy: 1.000 Validation accuracy: 0.995

Результаты показывают, что использование технологии регуляризации может лучше решить проблему переобучения модели, но параметры необходимо внимательно корректировать.

4.4. Реализация Dropout

Мы ввели Dropout для двух нейронных сетей и отключили нейроны в 25 % и 50 % после первого всесвязанного слоя соответственно. Для `ConvNN_1` мы прекратили обучение после 40-й итерации, чтобы избежать переобучения модели, а `ConvNN_2` 50 раз, и получены следующие результаты.

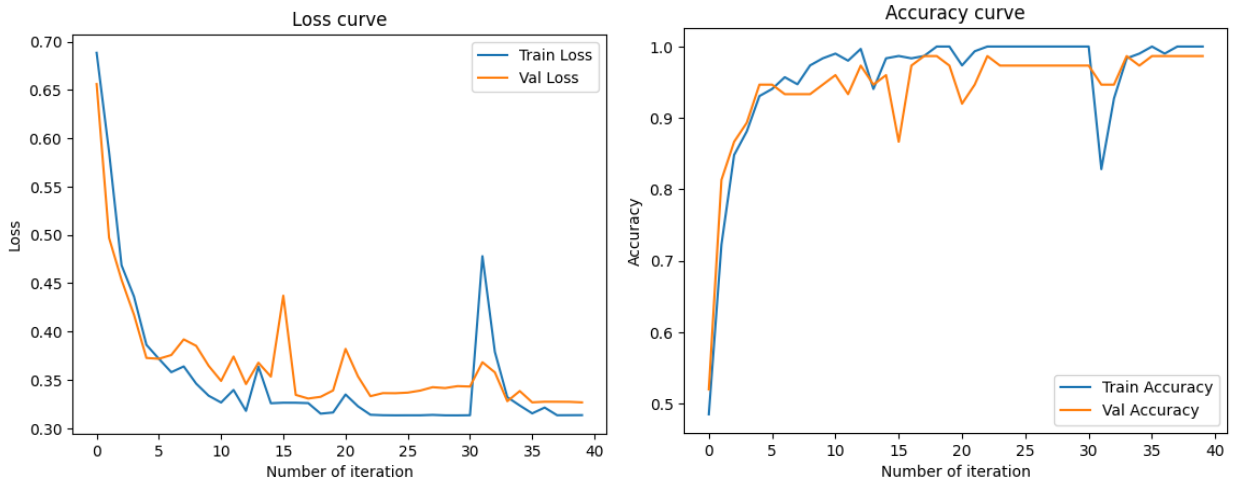

Рис.7 Loss и Accuracy моделя ConvNN_1 (25% Dropout)

Training loss: 0.324 Validation loss: 0.341 Training accuracy: 0.987 Validation accuracy: 0.971

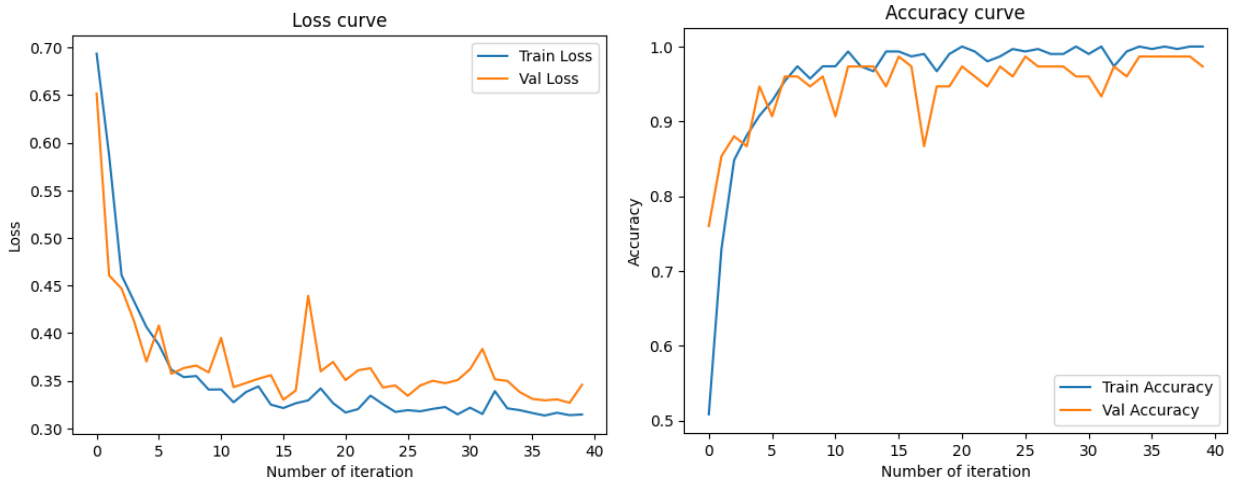

Рис.8 Loss и Accuracy моделя ConvNN_1 (50% Dropout)

Training loss: 0.315 Validation loss: 0.333 Training accuracy: 0.999 Validation accuracy: 0.984

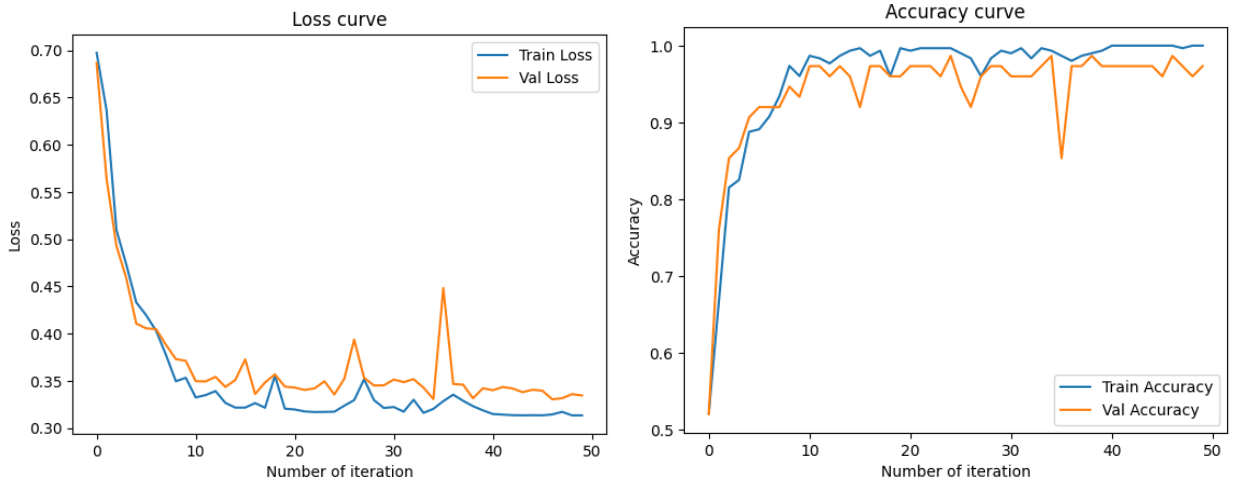

Рис.9 Loss и Accuracy моделя ConvNN_2 (25% Dropout)

Training loss: 0.314 Validation loss: 0.334 Training accuracy: 0.999 Validation accuracy: 0.971

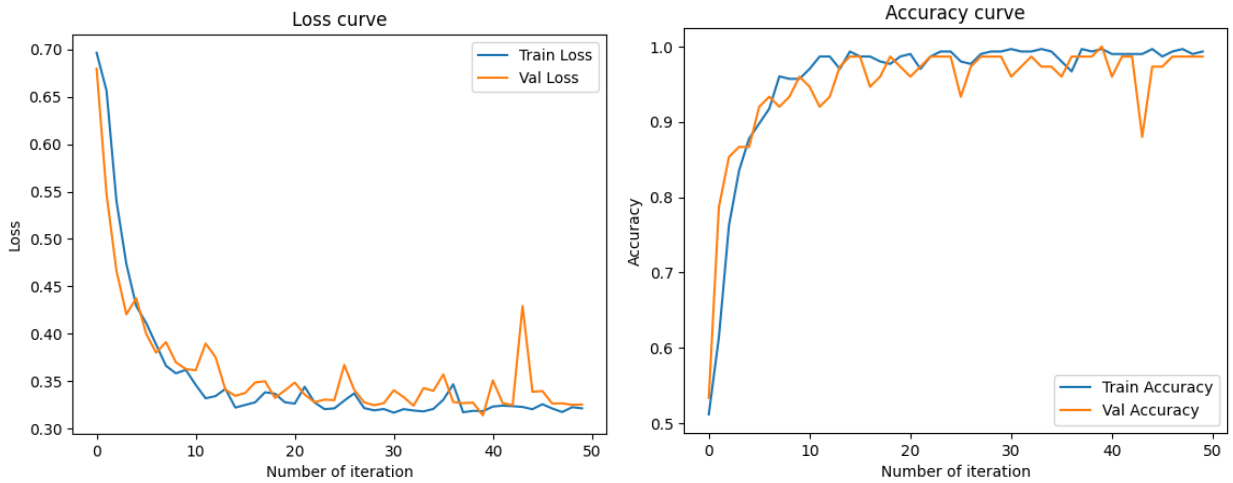

Рис.10 Loss и Accuracy моделя ConvNN_2 (50% Dropout)

Training loss: 0.322 Validation loss: 0.329 Training accuracy: 0.992 Validation accuracy: 0.984

Из приведенных выше результатов мы обнаружили, что производительность 50% Dropout лучше, чем производительность 25% Dropout для обеих сетей.

4.5. Окончательная валидация и обучение

Наконец, на основе нашего вышеприведенного анализа, 50% Dropout и соответствующие параметры регуляризации были использованы для двух моделей соответственно, без аргументации изображения, для окончательной валидации, выполняем 60 итераций.

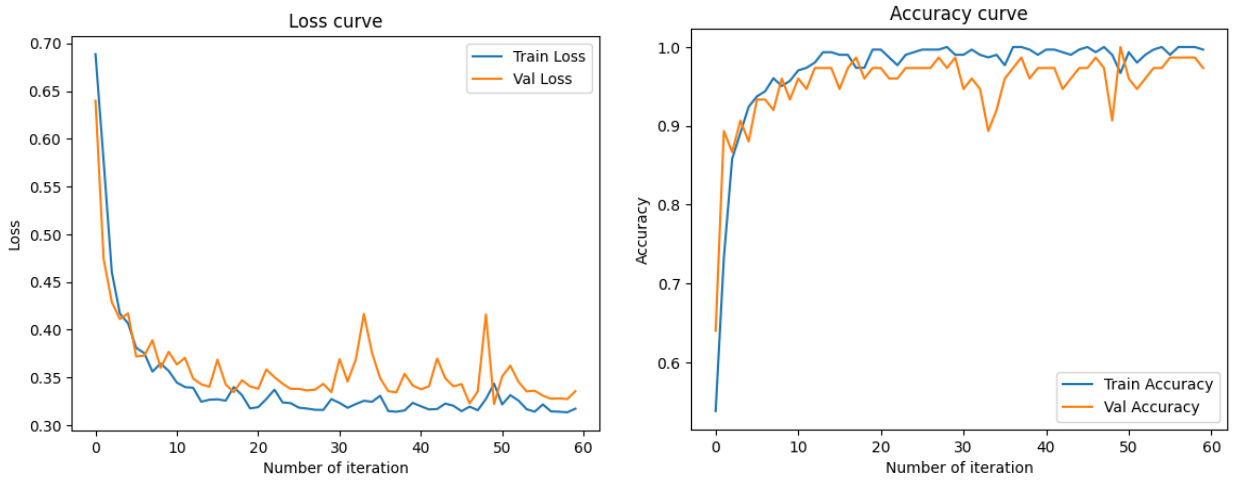

Рис.11 Loss и Accuracy моделя ConvNN_1 (Окончательная)

Training loss: 0.322 Validation loss: 0.339 Training accuracy: 0.993 Validation accuracy: 0.976

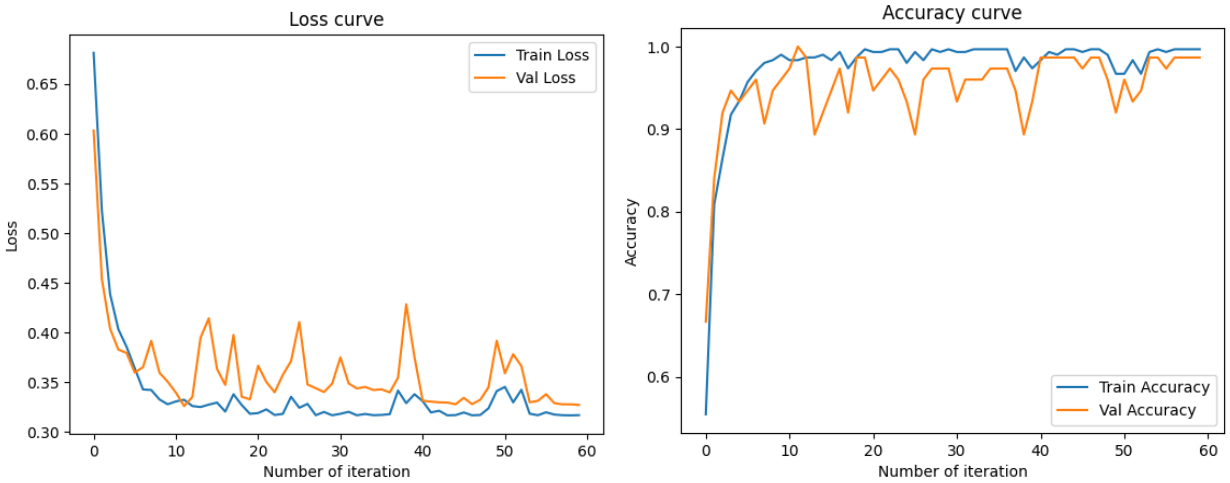

Рис.12 Loss и Accuracy моделя ConvNN_2 (Окончательная)

Training loss: 0.317 Validation loss: 0.330 Training accuracy: 0.996 Validation accuracy: 0.984

Наконец, мы используем весь обучающий набор и набор проверки для тестирования, сохраняем полученную модель и полагаемся на нее для прогнозирования.

V. ЗАКЛЮЧЕНИЕ

В этом исследовании мы обеспечиваем углубленное изучение обучения и оптимизации моделей нейронных сетей. Анализировали данные, применили методы регуляризации и изучили влияние Dropout, мы пришли к следующим выводам:

Проблема переобучения. Мы наблюдали значительные различия между обучающим и проверочным наборами, что указывает на то, что модель подвержена риску переобучения. Чтобы решить эту проблему, мы используем метод регуляризации L2, но для получения наилучших результатов необходимо настроить параметры.

Увеличение данных. Хотя в некоторых случаях увеличение данных может быть полезно при обучении модели, в этом конкретном наборе данных эффект был ограниченным. Мы добавили перевернутые по горизонтали изображения в качестве способа дополнения данных, но существенно не улучшили производительность модели.

Влияние отсева: мы изучили влияние различных коэффициентов отсева на обучение нейронной сети. Результаты показывают, что 50% отсева лучше, чем 25%, что имеет решающее значение для производительности модели.

В совокупности наше исследование дает полезную информацию об обучении и оптимизации моделей нейронных сетей. В дальнейшей работе возможно дальнейшее изучение других методов регуляризации, а также более сложных сетевых архитектур для повышения производительности модели.

БИБЛИОГРАФИЯ

[1] Yang, S.; Xiao, W.; Zhang, M.; Image Data Augmentation for Deep Learning: A Survey. arXiv:2204.08610v2, Nov 2023

[2] Ekachai Phaisangittisagul. Department of Electrical Engineering, Faculty of Engineering Kasetsart University Bangkok, Thailand. An Analysis of the Regularization between L2 and Dropout in Single Hidden Layer Neural Network. 2166-0670/16 2016 IEEE DOI 10.1109/ISMS.2016.14

[3] Srivastava, N.; Hinton, G.; Krizhevsky A.; Department of Computer Science University of Toronto. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15 (2014) 1929-1958

[4] Cross Validation in Machine Learning [Online]. Available: https://www.geeksforgeeks.org/cross-validation-machine-learning [Accessed: 29-MAR-2024].

[5] “Python 3.11.8 documentation,” Python Software Foundation. [Online]. Available: https://docs.python.org/3.11/. [Accessed: 29-MAR-2024].

[6] “Pytorch documentation” The Linux Foundation. [Online]. Available: https://pytorch.org/. [Accessed: 29-MAR-2024].

[7] CUDA Toolkit - Free Tools and Training, NVIDIA Developer. [Online]. Available: https://developer.nvidia.com/cuda-toolkit. [Accessed: 29-MAR-2024].

[8] CUDA Deep Neural Network (cuDNN), NVIDIA Developer. [Online]. Available: https://developer.nvidia.com/cudnn. [Accessed: 29-MAR-2024].