УДК 004.423

РАЗРАБОТКА И ОБУЧЕНИЕ МОДЕЛИ СВЁРТОЧНОЙ НЕЙРОСЕТИ ДЛЯ ИДЕНТИФИКАЦИИ ЛИЧНОСТИ ЧЕЛОВЕКА

Кулинча П.В.
студент направления подготовки информатика и вычислительная техника,
Хакасский государственный университет имени Н.Ф. Катанова,
г. Абакан, Россия [1]

Аннотация: В статье рассматривается процесс создания и обучения свёрточной нейросети для задачи идентификации личности по изображениям лица, включая архитектуру модели, этапы подготовки данных и визуализацию результатов обучения.

Ключевые слова: свёрточная нейросеть, идентификация личности, классификация изображений, обучение модели, архитектура сети, точность, потеря, аугментация.

DEVELOPMENT AND TRAINING OF A CONVOLUTIONAL NEURAL NETWORK MODEL TO IDENTIFY A PERSON'S IDENTITY

Kulincha P.V.
student of computer science and computer engineering department,
N.F. Katanov Khakass State University,
Abakan, Russia

Abstract: This paper presents the development and training process of a convolutional neural network for facial identity recognition, covering model architecture, data preparation steps, and visualization of training results.

Keywords: convolutional neural network, identity recognition, image classification, model training, network architecture, accuracy, loss, augmentation.

[1] Научный руководитель: Спирин Д.В. доцент кафедры ЦТиД, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия

Введение

Распознавание личности по изображениям лица является одной из ключевых задач компьютерного зрения и находит широкое применение в системах безопасности, биометрии, аутентификации и пользовательских интерфейсах. В последние годы значительный прогресс в этой области обеспечили свёрточные нейросети (CNN, Convolutional Neural Networks), которые способны эффективно извлекать сложные визуальные признаки и обучаться на больших объемах изображений.

Целью данной работы является разработка и обучение модели свёрточной нейросети, способной классифицировать лица и определять принадлежность к конкретному пользователю. В статье рассматриваются как теоретические аспекты построения архитектуры сети, так и практические шаги по её реализации, включая подготовку данных, применение аугментации, настройку параметров обучения и анализ результатов.

Теоретические основы свёрточных нейросетей

Свёрточные нейросети представляют собой разновидность искусственных нейронных сетей, предназначенных для обработки изображений и других данных с пространственной структурой. Их ключевое преимущество заключается в способности автоматически извлекать важные признаки из изображений без необходимости ручной инженерии признаков.

Основные компоненты CNN:

- **Сверточные слои (Conv2D):** Эти слои выполняют операцию свёртки — сканирование изображения фильтрами (ядрами), которые «выявляют» локальные шаблоны, такие как края, текстуры и углы. Каждый фильтр обучается находить свой уникальный признак.
- **Функции активации (ReLU):** Применяются после свертки для внесения нелинейности в модель. ReLU (Rectified Linear Unit) является стандартной функцией активации и позволяет эффективно решать задачи классификации.

- **Субдискретизация (Pooling):** Обычно используется слой максимального пуллинга (MaxPooling), который уменьшает размерность признаков, снижая вычислительные затраты и позволяя модели концентрироваться на наиболее выраженных признаках.
- **Полносвязные слои (Dense):** На заключительном этапе CNN-признаки преобразуются в вектор, который подаётся на один или несколько полносвязных слоев для финального принятия решения — например, к какому классу принадлежит изображение.
- **Dropout:** Механизм регуляризации, уменьшающий переобучение. В процессе обучения случайным образом «отключаются» некоторые нейроны, что заставляет сеть не полагаться только на ограниченное количество признаков.

Благодаря своей способности к иерархическому извлечению признаков, свёрточные нейросети стали стандартом де-факто в задачах компьютерного зрения, включая распознавание лиц, идентификацию объектов, сегментацию изображений и многое другое [1].

На рисунке 1 представлена типовая модель свёрточной нейросети [2].

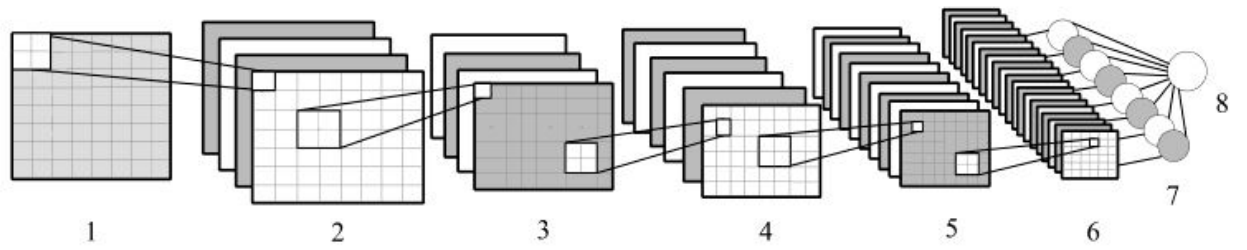

Рис. 1. *Архитектура свёрточной нейронной сети: 1 – вход; 2, 4, 6 – сверточные слои; 3, 5 – подвыборочные слои; 7 – слой из обычных нейронов; 8 – выход*

Рисунок 1 – Типовая модель свёрточной нейросети

Архитектура модели

Для решения задачи идентификации личности была разработана глубокая сверточная нейросеть, построенная с использованием API Keras. Модель имеет последовательную (Sequential) структуру и включает в себя чередование

сверточных и подвыборочных слоев (Pooling), что является классическим подходом к построению CNN для задач классификации изображений.

Сначала изображение размером 150×150×3 (цветное RGB) поступает на вход сверточному слою с 32 фильтрами, далее — слои с 64, 128 и 256 фильтрами. Каждый сверточный слой сопровождается операцией макспулинга — подвыборки, которая снижает размерность признакового пространства и способствует обобщающей способности модели (рисунок 2).

```
# Создаем модель
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(128, (3, 3), activation='relu'),  # Добавляем больше фильтров
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(256, (3, 3), activation='relu'),  # Ещё больше фильтров
    layers.MaxPooling2D((2, 2)),

    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dropout(0.5),  # Добавляем Dropout для предотвращения переобучения
    layers.Dense(29, activation='softmax')
])
```

Рисунок 2 – Описание модели нейросети на языке программирования Python [разработано автором]

Разрабатываемая модель свёрточной нейросети использует архитектуру типа Sequential, состоящую из четырёх блоков свёртки и подвыборки. Каждый блок начинается с слоя Conv2D, извлекающего признаки из изображения при помощи обучаемых фильтров, и завершается слоем MaxPooling2D, который уменьшает размерность данных и помогает выделить наиболее значимые признаки. По мере углубления модели увеличивается число фильтров: от 32 до 256, что позволяет улавливать сначала простые границы и формы, а затем — более сложные элементы изображения, включая особенности лица.

После блока свёрток данные преобразуются в одномерный вектор (Flatten), который подаётся на полносвязный слой из 512 нейронов с активацией ReLU. Для борьбы с переобучением используется слой Dropout с вероятностью отключения нейронов 50%. Выходной слой состоит из 29 нейронов с активацией softmax — каждый соответствует одной из классовых меток (личностей), и возвращает вероятностное распределение, по которому модель определяет наиболее вероятную личность на изображении.

Модель компилируется с использованием оптимизатора **Adam**, функции потерь categorical_crossentropy и метрики accuracy (рисунок 3).

```
# Компиляция модели
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Рисунок 3 – Компиляция модели нейросети на языке программирования Python [разработано автором]

Оптимизатор Adam (Adaptive Moment Estimation) является одним из наиболее популярных и эффективных алгоритмов для оптимизации нейросетей, который сочетает в себе лучшие черты двух других методов: AdaGrad и RMSProp. Он адаптирует скорость обучения для каждого параметра модели, используя оценки первого и второго моментов градиента [3].

Обучение модели

После того как модель сверточной нейросети была создана и подготовлена, следующим этапом является её обучение. Обучение модели состоит в минимизации функции потерь с использованием оптимизатора, который подбирает параметры сети (веса) таким образом, чтобы минимизировать ошибку на обучающих данных.

В коде для обучения модели используется метод `fit()`, который запускает процесс тренировки, где в качестве данных подаются изображения и метки классов из генераторов `train_generator` и `validation_generator`. Генератор обучающих данных обеспечивает подачу изображений в пакетах (батчах) по 32 изображения, что позволяет эффективно использовать память и ускоряет

процесс обучения. Процесс тренировки проходит по нескольким эпохам (в данном случае 50), что позволяет модели улучшать свою точность по мере обработки большего числа примеров (рисунок 4).

```
# Обучение модели (увеличиваем количество эпох)
history = model.fit(
    train_generator,
    epochs=50,  # Увеличиваем число эпох
    validation_data=validation_generator
)
```

Рисунок 4 – Обучение модели нейросети на языке программирования Python [разработано автором]

Кроме того, в процессе обучения автоматически отслеживаются два показателя: точность (accuracy) и потери (loss) на обучающих и валидационных данных. Это позволяет контролировать переобучение и убедиться, что модель не подгоняет свои параметры исключительно под тренировочные данные, а умеет обобщать информацию на новые данные. По ходу обучения модель постепенно улучшает свою способность классифицировать изображения, минимизируя потери и повышая точность. На рисунке 5 представлен процесс обучения нейросети.

```
Epoch 40/50
9/9 ━━━━━━━━━━━━━━━━━━━━ 24s 3s/step - accuracy: 0.7455 - loss: 0.8344 - val_accuracy: 0.7963 - val_loss: 0.6119
Epoch 41/50
9/9 ━━━━━━━━━━━━━━━━━━━━ 28s 3s/step - accuracy: 0.7563 - loss: 0.8626 - val_accuracy: 0.8148 - val_loss: 0.6856
Epoch 42/50
9/9 ━━━━━━━━━━━━━━━━━━━━ 24s 3s/step - accuracy: 0.7176 - loss: 0.9691 - val_accuracy: 0.8148 - val_loss: 0.5390
Epoch 43/50
9/9 ━━━━━━━━━━━━━━━━━━━━ 24s 3s/step - accuracy: 0.7226 - loss: 0.8653 - val_accuracy: 0.8704 - val_loss: 0.3706
Epoch 44/50
9/9 ━━━━━━━━━━━━━━━━━━━━ 24s 3s/step - accuracy: 0.7861 - loss: 0.7486 - val_accuracy: 0.8704 - val_loss: 0.4877
Epoch 45/50
9/9 ━━━━━━━━━━━━━━━━━━━━ 41s 3s/step - accuracy: 0.7528 - loss: 0.7860 - val_accuracy: 0.8333 - val_loss: 0.4498
Epoch 46/50
9/9 ━━━━━━━━━━━━━━━━━━━━ 23s 3s/step - accuracy: 0.7215 - loss: 0.7668 - val_accuracy: 0.8333 - val_loss: 0.4887
Epoch 47/50
9/9 ━━━━━━━━━━━━━━━━━━━━ 23s 3s/step - accuracy: 0.7385 - loss: 0.8639 - val_accuracy: 0.8519 - val_loss: 0.4674
Epoch 48/50
9/9 ━━━━━━━━━━━━━━━━━━━━ 25s 3s/step - accuracy: 0.7819 - loss: 0.7227 - val_accuracy: 0.8148 - val_loss: 0.6323
Epoch 49/50
9/9 ━━━━━━━━━━━━━━━━━━━━ 23s 3s/step - accuracy: 0.7812 - loss: 0.6645 - val_accuracy: 0.8889 - val_loss: 0.3885
Epoch 50/50
9/9 ━━━━━━━━━━━━━━━━━━━━ 23s 3s/step - accuracy: 0.7640 - loss: 0.6821 - val_accuracy: 0.8333 - val_loss: 0.5854
```

Рисунок 5 – Процесс обучения нейросети [разработано автором]

Визуализация графиков точности и потерь позволяет наглядно наблюдать прогресс обучения модели. Эти графики помогают понять, как модель учится, и есть ли проблемы с переобучением (например, если точность на валидационных

данных перестает улучшаться или даже ухудшается). Результаты на графиках, отображающие точность и потери по эпохам, дают полезную информацию о стабильности обучения и эффективности модели (рисунок 6).

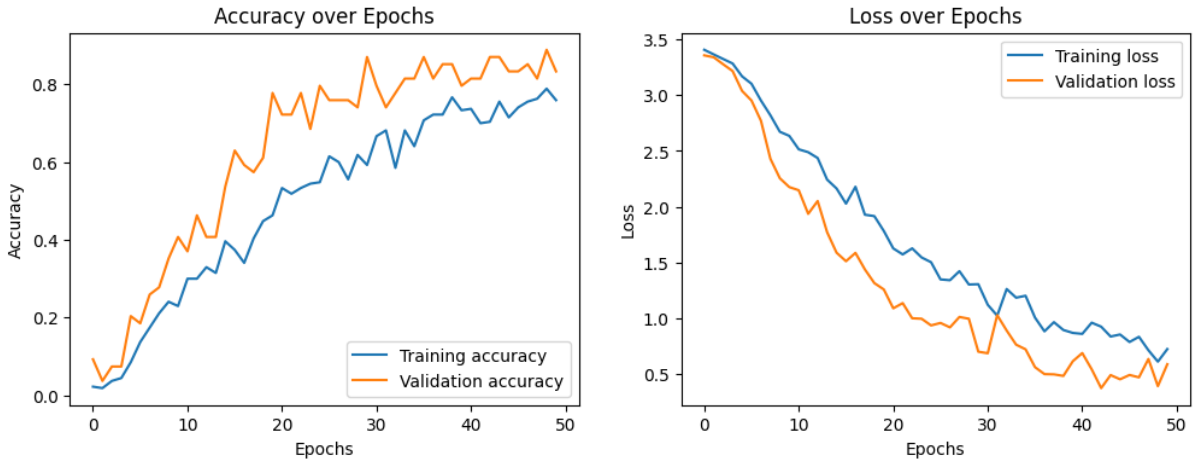

Рисунок 6 – Визуализация графиков точности и потерь при обучении нейросети [разработано автором]

После завершения обучения модель сохраняется для дальнейшего использования, что позволяет применять её для классификации новых изображений. В данном коде модель сохраняется с помощью метода model.save(), и файл модели сохраняется в формате .h5, что является стандартным для сохраненных моделей TensorFlow/Keras (рисунок 7).

```
# Сохраняем модель
model.save('/content/face_model1_3.h5')
```

Рисунок 7 – Сохранение модели нейросети [разработано автором]

После обучения модели, общий вес файла в формате .h5 составил 460 Мбайт.

Результаты обучения и проверка работоспособности модели

После завершения 50 эпох обучения, модель показала следующие результаты: на обучающих данных точность составила **76.4%**, а на валидационных данных — **83.33%**. Потери на обучающих данных достигли **0.6821**, а на валидационных — **0.5854**. Эти показатели свидетельствуют о том, что модель успешно обучается и способна делать предсказания с высокой

точностью на валидационных данных, что указывает на её хорошую способность к обобщению.

Для более детальной оценки работоспособности модели были проведены дополнительные проверки с использованием изображений, не встречавшихся в процессе обучения. В одном из тестов, изображение человека по имени **Ewa** было подано на вход нейросети, которая, в свою очередь, предсказала, что на фото изображена **Ewa**, присвоив этому предсказанию уверенность **0.9998** (рисунки 8-9). Это результат высоко точного распознавания, который подтверждает, что нейросеть правильно идентифицирует личность, с минимальной вероятностью ошибки. Выходные данные каждого нейрона также были проанализированы, что позволило визуализировать, как нейросеть делает свои предсказания и оценивает вероятности для каждого класса.

```
Alejandra: 0.0000
Alessandro: 0.0000
Anastasia: 0.0000
Andrea_Ran: 0.0000
Anna: 0.0000
Bruno: 0.0000
Clarissa: 0.0000
Daiane: 0.0000
Diego: 0.0002
Ewa: 0.9998
Fernanda: 0.0000
Gabriel: 0.0000
GuC,PґP°+Pєlten_CC,PґP°TB¦ayC,P¤PђC,P¦PўrcC,P¤PђC,P¦Pў: 0.0000
Juliana: 0.0000
Kasia: 0.0000
Kateryna: 0.0000
Klara: 0.0000
Luis: 0.0000
Mark: 0.0000
Massimiliano: 0.0000
Matheus: 0.0000
Miia: 0.0000
Mykhailo: 0.0000
Paolo: 0.0000
Rayanne: 0.0000
RoC,PґP°+P№mulo: 0.0000
Valeriia: 0.0000
Vitalijs: 0.0000
Weslley: 0.0000
```

Рисунок 7 – Значения выходных нейронов [разработано автором]

Рисунок 8 – Предсказание личности человека [разработано автором]

Этот результат подтверждает, что обученная модель эффективно работает на реальных данных и способна точно распознавать лица, делая правильные предсказания с высокой уверенностью.

Библиографический список:

1. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems (NIPS), 25, 1097–1105.

2. Технологии распознавания лиц или фейсконтроль по-умному | iot.ru Новости Интернета вещей [Электронный ресурс] – URL: https://iot.ru/gorodskaya-sreda/tekhnologii-raspoznavaniya-lits-ili-feyskontrol-po-umnomu

3. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Оригинальность 81%