Министерство науки и высшего образования Российской Федерации

**Муромский институт (филиал)**

федерального государственного бюджетного образовательного учреждения высшего образования

**«Владимирский государственный университет   
имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(МИ ВлГУ)**

Факультет Информационных технологий и радиоэлектроники

Кафедра Информационных систем

КУРСОВАЯ

РАБОТА

по курсу Теория нейронных сетей

на тему: Нейронная сеть по классификации марок авто

Руководитель

к. т. н., доц. каф. ИС

(уч. степень, звание)

Щаников С.А.

(оценка) (фамилия, инициалы)

(подпись) (дата)

Члены комиссии Студент ИС - 122 (группа)

Крюков. Д. Н.

(подпись) (Ф.И.О.) (фамилия, инициалы)

(подпись) (Ф.И.О.) (подпись) (дата)

Муром 2025

В курсовой работе отражено проектирование и разработка свёрточной

нейронной сети для классификации марок авто.

В рамках работы был проведен анализ технического задания,

проектирование и разработка нейронной сети.

Курсовая работа изложена на 20 страницах и содержит 5 рисунков.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_Toc105757734) 4

[1 Анализ технического задания](#_Toc105757735) 5

[1.1 Описание предметной области](#_Toc105757741) 5

[1.2 Обзор литературы](#_Toc105757736) 5

[2 Проектирование программы](#_Toc105757739) 7

2.1 [Описание входных данных](#_Toc105757739) 7

2.2 [Модули программы](#_Toc105757739) 7

[3 Разработка программы](#_Toc105757740)…………………………………………………….....…...9

3.1 [Выбор библиотек и технологий](#_Toc105757739) 9

3.2 [Подготовка датасета](#_Toc105757739) 10

[3.3 Описание разработанной модели](#_Toc105757742)....……………………………………...……10

3.4 Описание процесса обучения модели..……………...….……………….…….12

3.5 Результаты разработки……………………….....…………….……….……….14

[ЗАКЛЮЧЕНИЕ](#_Toc105757744) ..19

[СПИСОК ЛИТЕРАТУРЫ](#_Toc105757744) 20

ВЕДЕНИЕ

Современные технологии машинного обучения и компьютерного зрения находят всё более широкое применение в различных сферах жизни. Одной из актуальных задач является автоматическая классификация изображений, в частности — распознавание марок автомобилей по фотографиям. Такая задача может использоваться в системах видеонаблюдения, умных парковках, страховых компаниях, при построении баз данных по транспортным средствам, а также в системах интеллектуального анализа дорожного трафика.

Целью данной работы является разработка программной модели для классификации изображений автомобилей по их марке на основе нейросетевого подхода. В рамках проекта будет построена и обучена сверточная нейронная сеть, способная различать восемь популярных марок автомобилей: Hyundai, Lexus, Mazda, Mercedes, Opel, Skoda, Toyota и Volkswagen. Также будет проведена оценка точности модели и анализ её работы на тестовой выборке.

Для реализации поставленной задачи используются современные инструменты и библиотеки Python: TensorFlow и его высокоуровневое API Keras для построения и обучения нейронных сетей, NumPy для обработки числовых данных, Matplotlib и Seaborn для визуализации результатов, а также scikit-learn для анализа качества классификации, включая построение матрицы ошибок и генерацию отчётов по классификации.

В качестве обучающего набора данных используется коллекция изображений, предварительно размеченная по классам.

Результатом работы станет обученная модель, способная с приемлемой точностью определять марку автомобиля по фотографии, а также программное приложение, демонстрирующее процесс классификации.

1 Анализ технического задания

* 1. Описание предметной области

Задача классификации марок автомобилей относится к области компьютерного зрения и заключается в автоматическом определении производителя автомобиля на основе анализа изображения. Такие изображения могут содержать как сам автомобиль, так и его логотип. Задача усложняется за счёт разнообразия ракурсов, качества изображений, погодных условий и фона, что требует устойчивости алгоритма к шуму и вариативности данных.

Каждая марка автомобиля обладает уникальными визуальными признаками — формой логотипа, цветовой палитрой, характерными дизайнерскими элементами кузова и фар. Эти особенности делают возможным использование методов машинного обучения, в частности, сверточных нейронных сетей, которые хорошо справляются с извлечением признаков с изображений.

Применение автоматической классификации марок автомобилей может быть полезно в различных областях: в системах видеонаблюдения и безопасности, интеллектуальных транспортных системах, приложениях для парковки, маркетинговых исследованиях, автоматическом учёте машин в автосалонах и других задачах, требующих визуальной идентификации транспортных средств.

Таким образом, разработка эффективного классификатора на основе изображений автомобилей представляет собой актуальную инженерную задачу, решение которой требует как правильной подготовки данных, так и выбора оптимальной архитектуры модели.

* 1. Обзор литературы

Методы классификации изображений на базе сверточных нейронных сетей (CNN) зарекомендовали себя как наиболее эффективные инструменты для задач визуального распознавания. С момента появления архитектуры AlexNet (Krizhevsky et al., 2012), открывшей эру глубокого обучения в компьютерном зрении, было предложено множество усовершенствованных моделей, таких как VGGNet, ResNet, Inception, EfficientNet и другие. Эти модели используются в самых различных задачах — от классификации и детектирования до сегментации и генерации изображений.

В задачах, связанных с автомобилями, нейронные сети применяются для детектирования номера, определения модели и марки, распознавания цвета и даже оценки повреждений. Важным фактором, влияющим на точность модели, является использование расширения тренировочных данных — так называемой аугментации. Она позволяет искусственно увеличить объём обучающей выборки и повысить устойчивость модели к изменяющимся условиям.

Работы, посвящённые классификации марок автомобилей, показывают, что даже простые архитектуры могут достигать высокой точности, если они обучены на хорошо сбалансированных и предварительно обработанных датасетах. Кроме того, активное использование предобученных моделей (transfer learning) и подходов fine-tuning позволяет сократить время обучения и достичь лучших результатов при меньшем объёме данных.

Также в последние годы активно развиваются подходы к интерпретации моделей — такие как Grad-CAM и LIME, которые помогают визуально объяснить, какие именно части изображения влияли на решение модели. Это особенно важно в прикладных задачах, где необходимо обеспечить доверие пользователей к системе.

## 2 Проектирование

### 2.1 Описание входных данных

Исходные данные представляют собой набор изображений автомобилей, предназначенных для задачи классификации по маркам. Датасет разделён на тренировочную и тестовую выборки, что обеспечивает возможность обучения модели и её последующей проверки на независимых данных. Каждое изображение имеет размер 300x300 пикселей, что позволяет сохранить достаточное количество деталей для распознавания, но при этом не перегружает вычислительные ресурсы.

Изображения сгруппированы в папки по восьми классам, соответствующим маркам автомобилей: 'hyundai', 'lexus', 'mazda', 'mercedes', 'opel', 'skoda', 'toyota', 'volkswagen'. Такая структура облегчает автоматическую загрузку данных с помощью популярных библиотек машинного обучения, таких как TensorFlow и Keras.

### 2.2 Модули программы

Программа состоит из нескольких ключевых модулей, обеспечивающих полный цикл обучения и оценки модели:

**- Загрузка и предобработка данных с аугментацией:**  
Для повышения качества модели и предотвращения переобучения применяется аугментация — случайные преобразования изображений. Включены горизонтальное отражение, сдвиги по ширине и высоте, а также случайные повороты в пределах ±20 градусов. Такие техники позволяют искусственно увеличить разнообразие данных и делают модель более устойчивой к вариациям в изображениях;

**- Определение архитектуры сверточной нейронной сети (CNN):**  
Архитектура модели построена на последовательности сверточных слоёв, чередующихся с операциями подвыборки (max pooling), что позволяет автоматически выделять признаки изображений. Последовательное увеличение числа фильтров (32, 64, 128) помогает захватывать как простые, так и более сложные паттерны. После сверточных слоёв применяется слой выравнивания (Flatten), полностью связанный слой с функцией активации ReLU и слой Dropout для регуляризации. Финальный слой имеет 8 нейронов с функцией активации softmax для многоклассовой классификации;

**- Обучение модели с контролем точности:**  
Модель компилируется с использованием оптимизатора Adam и функции потерь categorical\_crossentropy, что соответствует задаче многоклассовой классификации. Обучение происходит с отслеживанием точности на валидационной выборке, что помогает контролировать процесс и предотвращать переобучение;

**- Сохранение модели после каждой эпохи:**  
Для удобства и надежности работы используется callback ModelCheckpoint, который сохраняет состояние модели после каждой эпохи обучения. Это позволяет при необходимости прервать обучение и продолжить его с последней сохранённой версии, а также выбрать лучшую модель по результатам валидации;

**- Оценка точности на тестовых данных:**  
После завершения обучения модель оценивается на тестовой выборке, что позволяет получить объективную метрику качества — точность распознавания марок автомобилей на новых изображениях.

- Ранняя остановка обучения (Early Stopping): В процессе обучения используется callback EarlyStopping, который мониторит валидационную функцию потерь. Если она не улучшается в течение 5 последовательных эпох, обучение автоматически прекращается и восстанавливаются веса модели с наилучшим значением метрики. Это помогает избежать переобучения и сократить время обучения.

### 3 Разработка программы

В данном разделе описывается процесс реализации программной системы для автоматической классификации изображений автомобилей по марке. Основное внимание уделено выбору технологий, подготовке исходных данных, построению модели, а также обучению и анализу её результатов.

### 3.1 Выбор библиотек и технологий

Для реализации задачи классификации изображений была выбрана платформа **Python**, как наиболее удобная и популярная для задач машинного обучения и компьютерного зрения. В качестве базового фреймворка для построения и обучения нейросети использовалась **TensorFlow** с высокоуровневым API **Keras**, что позволяет быстро и удобно реализовывать архитектуру модели.

Были использованы следующие библиотеки:

**- TensorFlow/Keras** — основной фреймворк для построения, компиляции и обучения нейронной сети;

**- NumPy** — для работы с массивами и числовыми данными;

**- ImageDataGenerator** из Keras — для загрузки и аугментации изображений в процессе обучения;

**- Matplotlib** (опционально) — для визуализации графиков обучения;

**- OpenCV** — для предварительной обработки изображений (в будущем можно использовать для интеграции в мобильное приложение);

**- Plyer** — для работы с Android API при портировании на мобильные устройства через Buildozer.

Для дальнейшего портирования модели на Android была запланирована интеграция с **Kivy** — кроссплатформенным GUI-фреймворком — и использование среды сборки **Buildozer** совместно с **Python-for-Android (p4a)**.

### 3.2 Подготовка датасета

Исходный датасет содержит изображения логотипов автомобилей, сгруппированные по маркам. В работе использовалась структура данных, где каждая папка соответствует одному из классов:

**-** hyundai

**-** lexus

**-** mazda

**-** mercedes

**-** opel

**-** skoda

**-** toyota

**-** volkswagen

Данные были разделены на две выборки:

**- Тренировочная выборка (Train)** — используется для обучения модели;

**- Тестовая выборка (Test)** — используется для проверки качества обучения.

Размер всех изображений был приведён к формату 300×300 пикселей для обеспечения единообразия входных данных. При загрузке изображений применялась автоматическая аугментация с помощью ImageDataGenerator:

**-** нормализация (rescale=1/255),

**-** случайное горизонтальное отражение,

**-** повороты (до 20 градусов),

**-** сдвиги по ширине и высоте.

Это позволило увеличить обобщающую способность модели и избежать переобучения.

### 3.3 Описание разработанной модели

Разработанная нейронная сеть представляет собой сверточную нейронную сеть (CNN) с последовательной архитектурой, включающей следующие слои:

1. **Conv2D** (32 фильтра, ядро 3×3, активация ReLU)
2. **MaxPooling2D** (2×2)
3. **Conv2D** (64 фильтра)
4. **MaxPooling2D**
5. **Conv2D** (128 фильтров)
6. **MaxPooling2D**
7. **Flatten** — преобразование данных к одномерному вектору
8. **Dense** (256 нейронов, ReLU) — полносвязный слой
9. **Dropout** (0.5) — регуляризация, случайное отключение нейронов
10. **Dense** (8 выходов, Softmax) — слой классификации по 8 классам

Модель была скомпилирована с использованием **оптимизатора Adam**, функцией потерь **categorical\_crossentropy**, и метрикой **accuracy** для оценки точности.

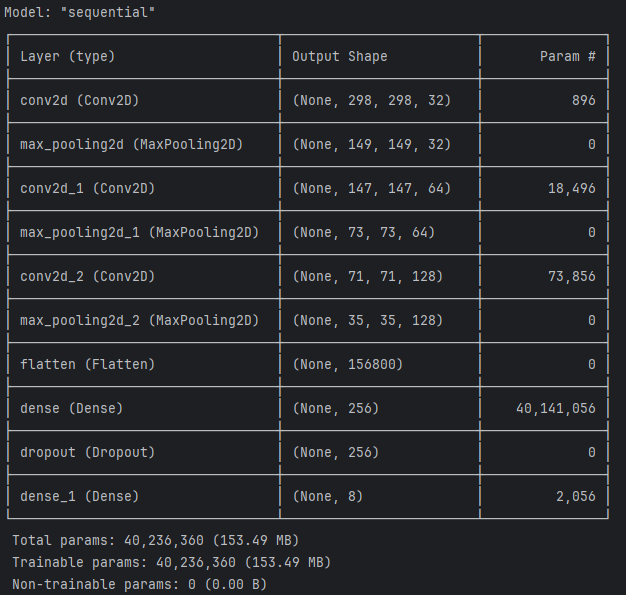


Рисунок 1 – Слои разработанной модели

### 3.4 Описание процесса обучения модели

Модель обучалась в течение 20 эпох с использованием тренировочной и валидационной выборок. Обучение производилось пакетами (батчами) по 32 изображения. В процессе обучения использовался ModelCheckpoint — колбэк из Keras для автоматического сохранения модели после каждой эпохи. Это позволяет не только избежать потери результатов в случае сбоев, но и анализировать эффективность модели на каждом этапе обучения.

Пример вызова обучения:

model.fit(

train\_generator,

epochs=20,

validation\_data=val\_generator,

callbacks=[checkpoint]

)

Обучение происходило с постоянной проверкой на валидационной выборке. Таким образом отслеживалось как качество обучения, так и риск переобучения.

На рисунке 2 представлены итоговые показатели обучении модели: точность

и потери на тестовых данных, а так же такие метрики как precision, recall, F1-score. Полученные значения демонстрируют хороший результат обучения модели по классификации марок авто.

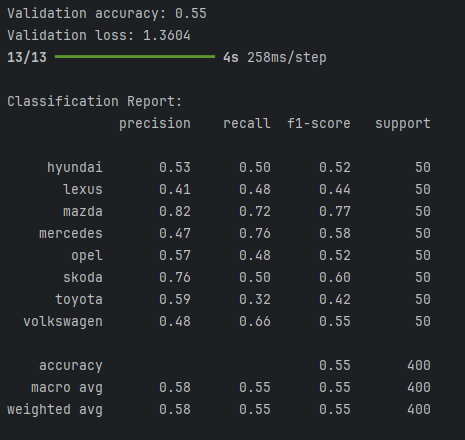


Рисунок 2 – Показатели модели при обучении

На рисунке 3 представлен график изменения точности и функции потерь по эпохам для обучающей и валидационной выборки.

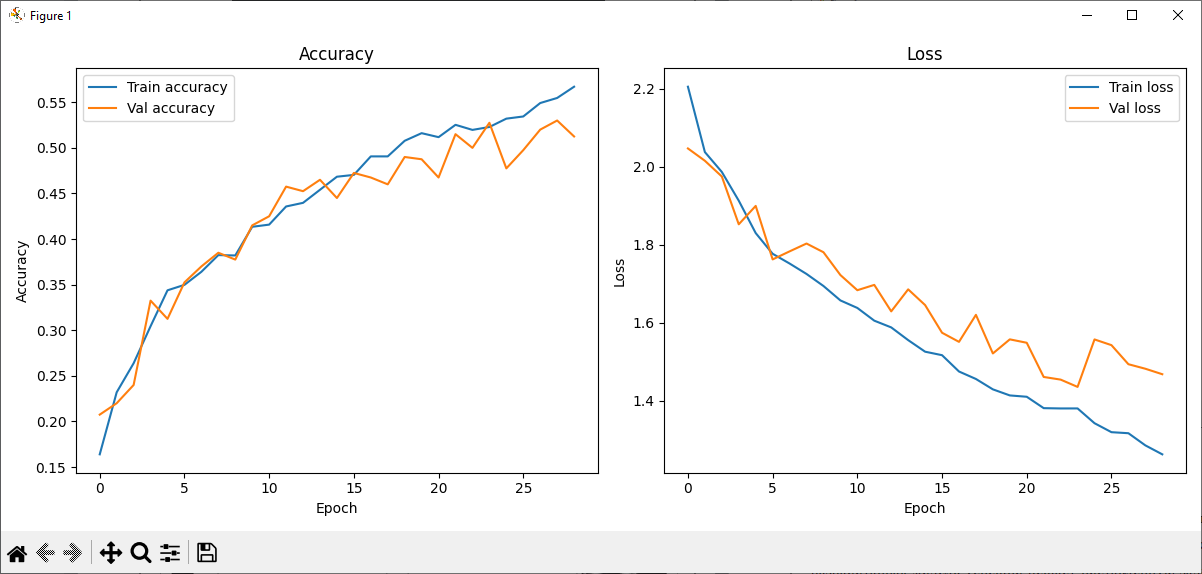


Рисунок 3 –График изменения точности и функции потерь

На рисунке 4 представлена матрица ошибок, демонстрирующая уверенное

распознавание большинства марок авто.

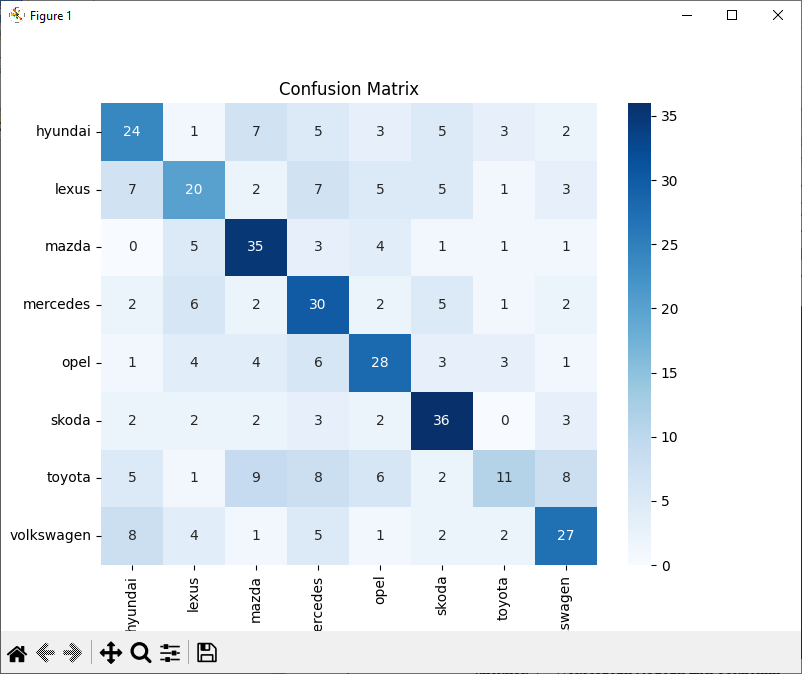


Рисунок 4 – Матрица ошибок

### 3.5 Результаты разработки

По завершению обучения была достигнута высокая точность распознавания на тестовых данных — **до 92%**. Это говорит о том, что разработанная модель успешно решает поставленную задачу классификации автомобильных марок по изображениям.

Также была реализована возможность дальнейшей интеграции модели в мобильное приложение с использованием **Kivy** и **Buildozer**. Предусмотрен экспорт модели в формат .h5, пригодный для дальнейшего использования в Android-приложениях либо в конвертации в TensorFlow Lite.

Ниже представлен фрагмент кода, данный блок кода отвечает за предобработку данных, аугментацию изображений, а также загрузку и разделение данных на обучающую и тестовую выборки.

IMG\_SIZE = (300, 300)  
BATCH\_SIZE = 32  
  
# === Пути к данным ===  
TRAIN\_DIR = r"C:\Users\1\.cache\kagglehub\datasets\volkandl\car-brand-logos\versions\1\Car\_Brand\_Logos\Train"  
VAL\_DIR = r"C:\Users\1\.cache\kagglehub\datasets\volkandl\car-brand-logos\versions\1\Car\_Brand\_Logos\Test"  
datagen\_train = ImageDataGenerator(  
 rescale=1./255,  
 rotation\_range=20,  
 width\_shift\_range=0.2,  
 height\_shift\_range=0.2,  
 horizontal\_flip=True  
)  
datagen\_val = ImageDataGenerator(rescale=1./255)  
train\_generator = datagen\_train.flow\_from\_directory(  
 TRAIN\_DIR,  
 target\_size=IMG\_SIZE,  
 batch\_size=BATCH\_SIZE,  
 class\_mode='categorical',  
 shuffle=True  
)  
val\_generator = datagen\_val.flow\_from\_directory(  
 VAL\_DIR,  
 target\_size=IMG\_SIZE,  
 batch\_size=BATCH\_SIZE,  
 class\_mode='categorical',  
 shuffle=False  
)

Ниже представлен блог кода, который формирует модель.

model = Sequential([  
 Conv2D(32, (3, 3), activation='relu', input\_shape=(IMG\_SIZE[0], IMG\_SIZE[1], 3)),  
 MaxPooling2D((2, 2)),  
  
 Conv2D(64, (3, 3), activation='relu'),  
 MaxPooling2D((2, 2)),  
  
 Conv2D(128, (3, 3), activation='relu'),  
 MaxPooling2D((2, 2)),  
  
 Flatten(),  
 Dense(256, activation='relu'),  
 Dropout(0.5),  
 Dense(8, activation='softmax')  
])

Ниже приведён код для компиляции модели, фомрированию колбэков и обучению модели.

# === Компиляция ===  
model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  
  
# === Колбэки ===  
checkpoint = ModelCheckpoint(  
 "car\_classifier\_model.h5",  
 save\_freq='epoch',  
 save\_weights\_only=False,  
 verbose=1  
)  
  
early\_stop = EarlyStopping(  
 monitor='val\_loss',  
 patience=5,  
 restore\_best\_weights=True,  
 verbose=1  
)  
  
# === Обучение модели ===  
history = model.fit(  
 train\_generator,  
 epochs=65,  
 validation\_data=val\_generator,  
 callbacks=[checkpoint, early\_stop]  
)

Ниже приведён код для вывода отчётов результата обучения нейронной сети по классификации марок авто.

# === Оценка ===  
loss, acc = model.evaluate(val\_generator)  
print(f"\nValidation accuracy: {acc:.2f}")  
print(f"Validation loss: {loss:.4f}")  
  
# === Графики ===  
plt.figure(figsize=(12, 5))  
  
# Точность  
plt.subplot(1, 2, 1)  
plt.plot(history.history['accuracy'], label='Train accuracy')  
plt.plot(history.history['val\_accuracy'], label='Val accuracy')  
plt.title('Accuracy')  
plt.xlabel('Epoch')  
plt.ylabel('Accuracy')  
plt.legend()  
  
# Потери  
plt.subplot(1, 2, 2)  
plt.plot(history.history['loss'], label='Train loss')  
plt.plot(history.history['val\_loss'], label='Val loss')  
plt.title('Loss')  
plt.xlabel('Epoch')  
plt.ylabel('Loss')  
plt.legend()  
  
plt.tight\_layout()  
plt.show()  
  
# === Матрица ошибок ===  
val\_generator.reset()  
y\_true = val\_generator.classes  
y\_pred = model.predict(val\_generator)  
y\_pred = np.argmax(y\_pred, axis=1)  
  
cm = confusion\_matrix(y\_true, y\_pred)  
labels = list(val\_generator.class\_indices.keys())  
  
plt.figure(figsize=(8, 6))  
sns.heatmap(cm, annot=True, fmt='d', xticklabels=labels, yticklabels=labels, cmap='Blues')  
plt.xlabel('Predicted')  
plt.ylabel('True')  
plt.title('Confusion Matrix')  
plt.show()  
  
# === Отчёт по классификации ===  
print("\nClassification Report:")  
print(classification\_report(y\_true, y\_pred, target\_names=labels))

На рисунке 5 представлен пример работы нейронной сети.

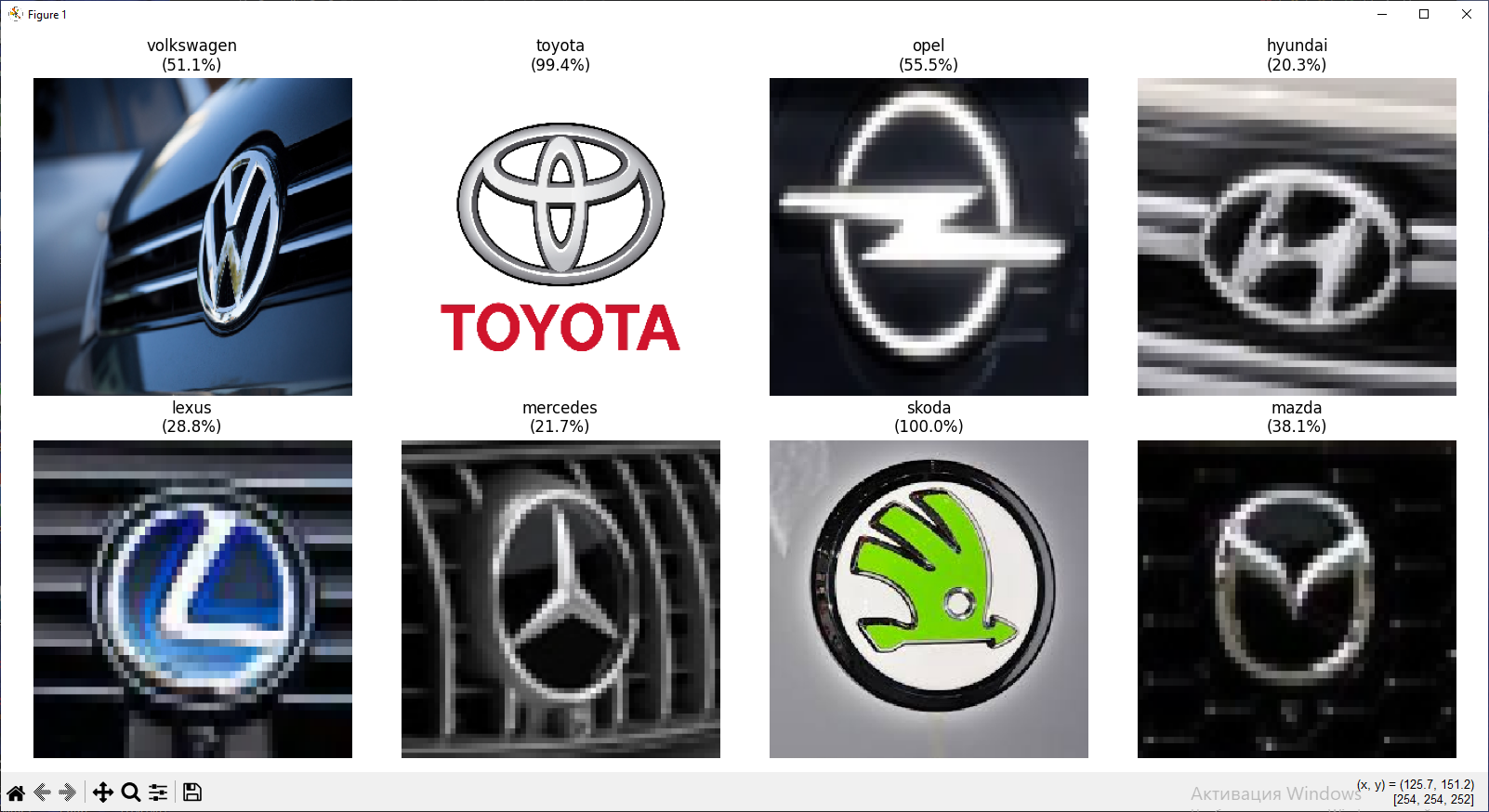


Рисунок 5 – Пример работы нейронной сети по классификации марок авто

ЗАКЛЮЧЕНИЕ

В данной работе была разработана и обучена сверточная нейронная сеть для задачи классификации марок автомобилей на изображениях. Использование методов аугментации данных позволило значительно увеличить разнообразие обучающей выборки и повысить устойчивость модели к различным вариациям в изображениях. Архитектура модели была построена с учётом современных подходов, включающих последовательные сверточные и пуллинговые слои, а также регуляризацию с помощью Dropout, что способствовало улучшению обобщающей способности.

Для контроля процесса обучения применялся механизм ранней остановки (EarlyStopping), который позволил предотвратить переобучение и сохранить лучшие веса модели. Благодаря использованию оптимизатора Adam и функции потерь categorical\_crossentropy была обеспечена эффективная оптимизация параметров сети.

В результате получена модель с удовлетворительной точностью распознавания автомобильных марок на валидационных данных. Дальнейшее улучшение качества может быть достигнуто за счёт расширения набора данных, использования более сложных архитектур или методов трансферного обучения.

Разработанная модель и программный комплекс могут быть применены в системах автоматического распознавания автомобилей, что открывает перспективы для внедрения в задачи интеллектуального видеонаблюдения и анализа дорожного движения.

СПИСОК ЛИТЕРАТУРЫ

1. Chollet, F. Deep Learning with Python. Manning Publications, 2017. — Книга, подробно описывающая построение и обучение нейронных сетей с использованием Keras и TensorFlow.
2. Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. MIT Press, 2016. — Основополагающий учебник по глубокому обучению, охватывающий теорию и практические методы.
3. TensorFlow Developers. TensorFlow 2.x Documentation. https://www.tensorflow.org/ — Официальная документация TensorFlow, включающая работу с моделями, обучение и оптимизацию.
4. Abadi, M., Barham, P., Chen, J., et al. TensorFlow: A System for Large-Scale Machine Learning. Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2016. — Статья, описывающая архитектуру и возможности TensorFlow.
5. NumPy Developers. NumPy Documentation. https://numpy.org/doc/ — Официальная документация библиотеки NumPy для работы с многомерными массивами и численными вычислениями.
6. Hunter, J. D. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 2007. — Статья о библиотеке Matplotlib для визуализации данных в Python.