Инженерный вестник Дона, №2 (2024)
ivdon.ru/ru/magazine/archive/n2y2024/8997

Разработка алгоритма переноса стиля изображения с использованием предобученной нейросети

Е.С. Майорова, Р.С. Зарипова

Казанский государственный энергетический университет, Казань

Аннотация: В данной статье представлен процесс разработки алгоритма, который способен извлекать стиль и содержание из двух разных изображений и создавать новое изображение, сохраняя структуру содержания одного изображения и одновременно применяя стилистические характеристики другого изображения. Данный алгоритм позволяет адаптировать стиль одного изображения к содержанию другого, создавая уникальные художественные произведения.
Ключевые слова: нейронные сети, перенос стиля, изображение, машинное обучение, алгоритм, набор данных, программное обеспечение.

Перенос стиля изображений – это задача, которая может быть решена с помощью алгоритмов машинного обучения [1, 2]. Для этого необходимо подготовить набор данных, состоящий из пар изображений, где одно изображение является исходным, а другое – целевым стилем, который нужно перенести на исходное изображение. Затем проводится обучение различных алгоритмов переноса стиля на этом наборе данных и сравнение их, используя метрики оценки качества переноса стиля, такие как среднеквадратическая ошибка или структурное сходство изображений [3, 4].

Целью данного исследования является разработка алгоритма, способного извлекать стиль и содержание из двух изображений и создавать новое изображение, сохраняя структуру содержания и одновременно применяя стилистические характеристики другого изображения. Это позволяет адаптировать стиль одного изображения к содержанию другого, создавая уникальные художественные произведения.

Научная новизна работы состоит в использовании алгоритма переноса стиля для создания новых изображений с высоким качеством и уникальным стилем.

Гипотеза исследования: при применении алгоритма переноса стиля изображений, основанного на машинном обучении, качество результирующего изображения будет выше, чем при использовании традиционных методов переноса стиля.

В соответствии с поставленной целью задачами исследования являются:

– анализ различных алгоритмов к переносу стиля изображений и их практическая реализация в библиотеках Python;

– исследование влияния параметров, таких как веса контента и стиля, на качество переноса стиля;

– оценка качества переноса стиля с помощью метрик оценки качества, которые позволяют сравнивать различные методы и алгоритмы переноса стиля;

– расширение возможностей переноса стиля, то есть анализ возможностей комбинирования переноса стиля с другими методами обработки изображений, такими как суперразрешение, удаление шума и изменение композиции.

Практическая значимость данного исследованиям заключается в возможности создания новых и оригинальных изображений, что может быть полезно в различных областях, таких, как дизайн, реклама, искусство, и т.д., а также применение при необходимости улучшении качества и эффективности процесса переноса стиля, что может привести к разработке новых инструментов и программного обеспечения для работы с изображениями [5].

Обучение нейросети включает в себя такие этапы:

1. Подготовка данных [6]. Это включает в себя сбор разнообразных изображений, которые будут использоваться в качестве контента и стиля для переноса. Также важно иметь пары изображений, где одно изображение будет исходным, а из второго будет переноситься стиль.

2. Выбор архитектуры модели машинного обучения.

3. Определение функции потерь для обучения модели [7].

4. Обучение модели машинного обучения путем минимизации функций потерь.

5. Тестирование для оценки производительности.

Результаты обучения могут зависеть от качества исходных данных, параметров модели и архитектуры [8, 9].

Описание набора данных

В нейронном переносе стиля используются два набора изображений:

– изображение контента – это изображение, стиль которого будет изменяться, сохраняя содержание изображения. Например, это может быть фотография пейзажа или портрета.

– изображение стиля – это изображение, стиль которого будет применяться к исходному изображению контента. Например, это может быть произведение искусства, такое, как картина Пикассо или Ван Гога, а также абстрактное изображение или что-то еще, характеризующееся определенным стилем, текстурой, цветами и другими характеристиками.

В данном решении будет использован архив images, содержащий в качестве изображений контента фотографии котов, а в качестве изображений стиля картины, исполненные в витражном стиле.

Начальные использованные фотографии показаны на рис. 1.

Подготовка набора данных

Подготовка изображений – это важный этап в задаче переноса стиля, который включает в себя несколько шагов для обработки и подготовки изображений стиля и содержания к использованию в алгоритме [10].

Первым шагом является распаковка архива с изображениями, а далее обрабатываются непосредственно сами изображения.

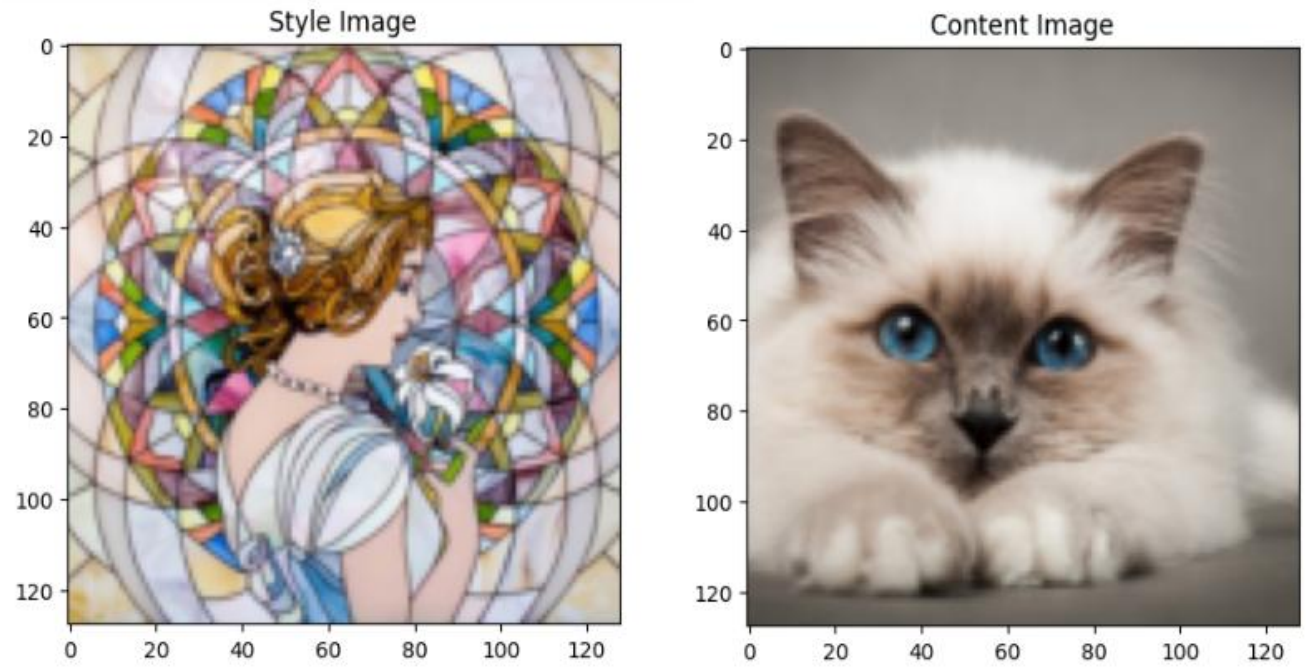

Рис. 1. – Исходные изображения

Через imsize определяется желаемый размер изображений в пикселях. В данном случае, размер равен 128x128 пикселей, что будет использовано для обоих изображений стиля и содержания.

Далее создается набор преобразований loader для обработки изображений. Этот набор включает в себя изменение размера изображения через transforms.Resize(imsize) до заданного значения (imsize), что обеспечивает одинаковый размер для обоих изображений стиля и содержания, обрезание центральной части изображения, чтобы сделать его квадратным (в данном случае 128x128), через transforms.CenterCrop(imsize) и преобразование изображения в тензор, который является форматом, используемым PyTorch для обработки изображений, через transforms.ToTensor().

Для удобства создана функция image_loader, которая загружает и обрабатывает изображения. Изображение открывается с использованием библиотеки PIL (Python Imaging Library) – функция Image.open(image_name),

а затем производятся преобразования, определенные в наборе loader – тензор создается из изображения и добавляется размерностью 1 для подготовки к передаче в PyTorch-модель – функция loader(image).unsqueeze(0).

Описание архитектур нейронных сетей

Для проведения исследования рассматривались следующие архитектуры нейронных сетей:

1. VGG19 (Visual Geometry Group 19).

Данная сеть состоит из 19 слоев. Слой Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) – первый сверточный слой с 64 фильтрами, размером ядра 3x3, паддингом 1, который принимает изображение с тремя каналами (RGB). После каждого сверточного слоя идёт слой активации ReLU. Далее идет второй сверточный слой с такими же параметрами, затем снова активация ReLU. После этого следует слой MaxPool2d(kernel_size=2, stride=2, padding=0), то есть пулинг слой, который уменьшает размер изображения в 2 раза (stride=2) с использованием операции максимума (max-pooling).

Этот процесс повторяется несколько раз, создавая последовательность сверточных слоев с различными количествами фильтров.

Архитектура VGG19 является важным инструментом при работе с изображениями. Помимо этого, выбор архитектуры VGG19 в качестве предобученной сети обусловлен такими факторами, как архитектурная сложность, что позволяет модели выделять и анализировать различные структуры и признаки в изображениях, а также предобучением на больших данных, так как данная сеть была предварительно обучена на больших наборах данных, таких, как ImageNet, который содержит миллионы изображений из тысячи категорий, что позволяет сети извлекать обобщенные признаки, которые могут быть полезными для различных задач обработки изображений [11].

2. Сверточная нейронная сеть.

Создание модели для переноса стиля осуществляется в функции get_style_model_and_losses. В этой функции модель составляется из предобученной модели VGG19 с добавлением слоев для измерения потерь стиля и контента.

Архитектура данной сети состоит из следующих компонент: слой нормализации Normalization, который приводит изображение к стандартным средним значениям и стандартным отклонениям. Это нестандартный слой, который не входит в стандартные сверточные сети, и его цель – нормализовать изображение перед подачей его на вход сети. Это важно для сохранения согласованных результатов при переносе стиля.

Далее следует слой Conv2d (3, 64, kernel_size = (3, 3), stride= (1, 1), padding= (1, 1)) – первый сверточный слой с 3 входными каналами (RGB) и 64 фильтрами. Размер ядра свертки – 3x3 с паддингом 1. Этот слой выполняет сверточное преобразование входного изображения.

StyleLoss представляет потерю стиля. В контексте переноса стиля он используется для измерения разницы между структурой извлеченных признаков текущего изображения и извлеченных признаков изображения стиля. Он помогает модели выучить структуру стиля.

После каждого сверточного слоя следует слой активации ReLU, который выполняет нелинейное преобразование. В данной архитектуре inplace=True, что означает, что вычисления выполняются на месте, и новое значение записывается в тот же тензор.

Пулинг-слой MaxPool2d, который уменьшает размер изображения в два раза с использованием операции максимума, что служит для уменьшения пространственных размеров изображения.

Архитектура данной сети повторяется для различных слоев (например, conv_2, conv_3, conv_4, conv_5) с разными параметрами, и каждый из них имеет свои потери стиля и потери содержания.

Код функции с созданием модели представлен на рис. 2, 3. На рис. 4 представлено описание полученной сети.

```
def get_style_model_and_losses(cnn, normalization_mean, normalization_std,
                               style_img, content_img,
                               content_layers=content_layers_default,
                               style_layers=style_layers_default):
    cnn = copy.deepcopy(cnn)

    normalization = Normalization(normalization_mean, normalization_std).to(device)

    content_losses = []
    style_losses = []

    model = nn.Sequential(normalization)

    i = 0
    for layer in cnn.children():
        if isinstance(layer, nn.Conv2d):
            i += 1
            name = 'conv_{}'.format(i)
        elif isinstance(layer, nn.ReLU):
            name = 'relu_{}'.format(i)
            layer = nn.ReLU(inplace=False)
        elif isinstance(layer, nn.MaxPool2d):
            name = 'pool_{}'.format(i)
        elif isinstance(layer, nn.BatchNorm2d):
            name = 'bn_{}'.format(i)
        else:
            raise RuntimeError('Unrecognized layer: {}'.format(layer.__class__.__name__))

        model.add_module(name, layer)
```

Рис. 2. – Код для создания функции

```
            if name in content_layers:
                target = model(content_img).detach()
                content_loss = ContentLoss(target)
                model.add_module("content_loss_{}".format(i), content_loss)
                content_losses.append(content_loss)

            if name in style_layers:
                target_feature = model(style_img).detach()
                style_loss = StyleLoss(target_feature)
                model.add_module("style_loss_{}".format(i), style_loss)
                style_losses.append(style_loss)

        for i in range(len(model) - 1, -1, -1):
            if isinstance(model[i], ContentLoss) or isinstance(model[i], StyleLoss):
                break

        model = model[:(i + 1)]

        return model, style_losses, content_losses
```

Рис. 3. – Код для создания функции (продолжение)

```
Sequential(
  (0): Normalization()
  (conv_1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (style_loss_1): StyleLoss()
  (relu_1): ReLU()
  (conv_2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (style_loss_2): StyleLoss()
  (relu_2): ReLU()
  (pool_2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv_3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (style_loss_3): StyleLoss()
  (relu_3): ReLU()
  (conv_4): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (content_loss_4): ContentLoss()
  (style_loss_4): StyleLoss()
  (relu_4): ReLU()
  (pool_4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv_5): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (style_loss_5): StyleLoss()
)
```

Рис. 4. – Описание полученной сети

Результат применения обученной модели

Нейросеть продемонстрировала способность к высококачественному переносу стиля, а полученное изображение является наглядным доказательством этого. Результат работы отображен на рис. 5.

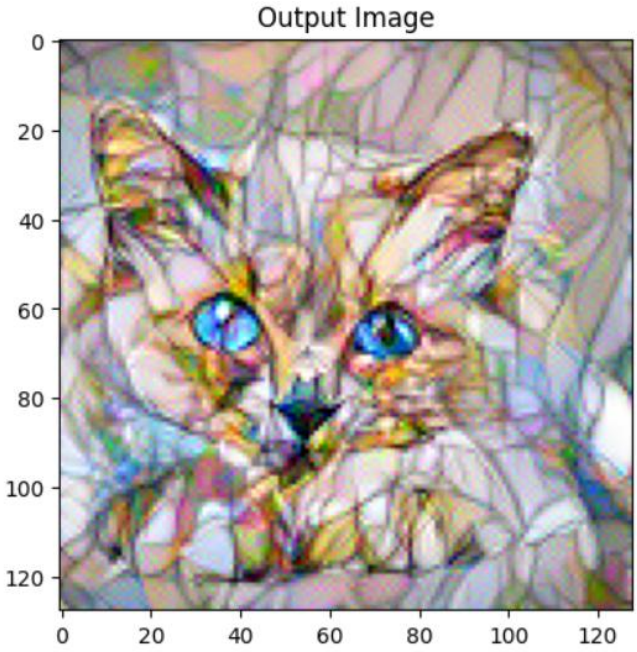

Рис. 5. – Результат работы нейросети

Таким образом, путем использования обученной нейронной сети создается модель, которая содержит все необходимые слои для выполнения переноса стиля, и она будет использоваться в дальнейшем в процессе оптимизации изображения. Данная архитектура нейросети используется для определения различий между содержанием и стилем изображений и последующего применения этих различий для создания новых изображений, которые сочетают в себе содержание одного изображения и стиль другого.

Литература

1. Силкина О.Ю., Зарипова Р.С. Тенденции в развитии искусственного интеллекта // Информационные технологии в строительных, социальных и экономических системах. 2020. № 3 (21). С. 63-65.

2. Алемасов Е.П., Зарипова Р.С. Перспективы применения технологий машинного обучения // Информационные технологии в строительных, социальных и экономических системах. 2020. № 2 (20). С. 32-34.

3. Емалетдинова Л.Ю., Кабирова А.Н., Катасев А.С. Методика разработки нейросетевых моделей регуляторов управления техническим объектом // Инженерный вестник Дона. 2023. № 7. URL: ivdon.ru/ru/magazine/archive/n7y2023/8544.

4. Менциев А.У., Айгумов Т.Г., Амирова Э.Ф. Методы и технологии сбора и анализа данных в цифровой экономике // Экономика: вчера, сегодня, завтра. 2022. Т. 12. № 11-1. С. 282-288.

5. Емалетдинова Л.Ю., Катасёв А.С., Назаров М.А. Нейронечеткая модель построения контуров на изображении // Инженерный вестник Дона. 2023. №7. URL: ivdon.ru/ru/magazine/archive/n7y2023/8535.

6. S.N. Cherny and R.F. Gibadullin. The Recognition of Handwritten Digits Using Neural Network Technology. 2022 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). 2022. pp. 965-970.

7. Пырнова О.А., Зарипова Р.С. Методы и проблемы переобучения многослойной нейронной сети // Информационные технологии в строительных, социальных и экономических системах. 2020. № 2 (20). С. 101-102.

8. Gizatullin Z.M., Gizatullin R.M., Nuriev M.G. Prediction of noise immunity of computing equipment under the influence of electromagnetic interference through the metal structures of building by physical modeling // Proceedings of the 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, EIConRus 2020. 2020. С. 120-123.

9. Овсеенко Г.А. SMART-решения и системы искусственного интеллекта // Информационные технологии в строительных, социальных и экономических системах. 2021. № 2 (24). С. 71-74.

10. Николаева С.Г., Ахунова И.Р. Интеграция SQL с технологиями блокчейн и искусственный интеллект // Современные цифровые технологии. Материалы II Всероссийской научно-практической конференции. Барнаул, 2023. С. 182-184.

11. Майорова Е.С., Зарипова Р.С. Решение задачи переноса стиля на изображения с использованием нейронных сетей // Научно-технический вестник Поволжья. 2023. № 11. С. 228-230.

References

1. Silkina O.YU., Zaripova R.S. Informacionnye tekhnologii v stroitel'nyh, social'nyh i ekonomicheskih sistemah. 2020. № 3 (21). pp. 63-65.

2. Alemasov E.P., Zaripova R.S. Informacionnye tekhnologii v stroitel'nyh, social'nyh i ekonomicheskih sistemah. 2020. № 2 (20). pp. 32-34.

3. Emaletdinova L.YU., Kabirova A.N., Katasev A.S. Inzhenernyj vestnik Dona. 2023. № 7. URL: ivdon.ru/ru/magazine/archive/n7y2023/8544.

4. Menciev A.U., Ajgumov T.G., Amirova E.F. Ekonomika: vchera, segodnya, zavtra. 2022. T. 12. № 11-1. pp. 282-288.

5. Emaletdinova L.YU., Katasyov A.S., Nazarov M.A. Inzhenernyj vestnik Dona. 2023. №7. URL: ivdon.ru/ru/magazine/archive/n7y2023/8535.

6. S.N. Cherny and R.F. Gibadullin. The Recognition of Handwritten Digits Using Neural Network Technology. 2022 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). 2022. pp. 965-970.

7. Pyrnova O.A., Zaripova R.S. Informacionnye tekhnologii v stroitel'nyh, social'nyh i ekonomicheskih sistemah. 2020. № 2 (20). pp. 101-102.

8. Gizatullin Z.M., Gizatullin R.M., Nuriev M.G. Proceedings of the 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, EIConRus 2020. 2020. pp. 120-123.

9. Ovseenko G.A. Informacionnye tekhnologii v stroitel'nyh, social'nyh i ekonomicheskih sistemah. 2021. № 2 (24). pp. 71-74.

10. Nikolaeva S.G., Ahunova I.R. Integraciya SQL s tekhnologiyami blokchejn i iskusstvennyj intellekt [Integrating SQL with blockchain and artificial intelligence technologies]. Sovremennye cifrovye tekhnologii. Materialy II Vserossijskoj nauchno-prakticheskoj konferencii. Barnaul, 2023. pp. 182-184.

11. Majorova E.S., Zaripova R.S. Nauchno-tekhnicheskij vestnik Povolzh'ya. 2023. № 11. pp. 228-230.

Дата поступления: 23.12.2023

Дата публикации: 1.02.2024