

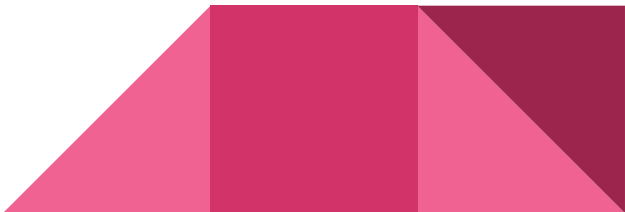
Определение времени на стрелочных часах

Бикбулатов Даниил, 407
Бабкина Анна, 407

Постановка задачи:

Создать нейронную сеть, способную по фотографии стрелочных часов определять время.

Предварительная работа:

- поиск датасета для тренировки модели + его улучшение
 - создание собственного датасета для тестирования полученной модели
 - выбор архитектуры нейронной сети
 - поиск оптимальных параметров для обучения
 - предобработка изображений
 - работа с библиотеками `fastai` и `timm`
- 

Структура проекта:

1. **clock_time_recognition** - основная программа в которой:
 - a. устанавливается предобученная модель
 - b. выбираются параметры модели
 - c. происходит обучение модели на предобработанных данных из датасета TIME - Image Dataset-Classification
 - d. обученная модель сохраняется
2. **clock_time_test** - программа, в которой загружается обученная модель и происходит тестирование обучения на собранном вручную наборе данных



timm

timm — это библиотека глубокого обучения, созданная Россом Вайтманом и представляющая собой набор SOTA моделей компьютерного зрения, слоев, утилит, оптимизаторов, планировщиков, загрузчиков данных, дополнений, а также сценариев обучения/проверки с возможностью воспроизведения результатов обучения ImageNet.

fastai

fastai - библиотека для глубокого обучения, представляющая собой надстройку над фреймворком pytorch.



Предобработка

- Изменение размера изображения [функция `Resize(128, method='squish')` из библиотеки `fastai` приводит картинку к размеру 128 x 128 с помощью метода `'squish'` (сжатие/ растяжение)]
- Случайное увеличение/кадрирование [функция `batch_tfms = aug_transforms(size=128, min_scale=0.75)`]
 - Картинка остаётся в размере 128 x 128
 - `aug_transforms` – аугментация данных, т.е. создание случайных вариаций исходных данных. Например, по умолчанию стоят: случайное переворачивание, вращение максимум на 10 градусов, максимальный зум 1.1, максимальная шкала изменения яркости 0.2, аффинное преобразование и изменение яркости с вероятностью 0.75. У нас также добавлен минимальный масштаб обрезки 0.75.
 - `batch_tfms` – применяем аугментацию ко всему пакету данных; используем видеокарту

Модель ConvNeXt

- предварительно обучена на Imagenet-22k и дополнительно настроенная на изображениях датасета Universal Image Embeddings (130 тыс, 128x128)
- base/ large/ xlarge - отвечают за размер модели
- 21841 классов

```
@register_model
def convnext_base_in22k(pretrained=False, **kwargs):
    model_args = dict(depths=[3, 3, 27, 3], dims=[128, 256, 512, 1024], **kwargs)
    model = _create_convnext('convnext_base_in22k', pretrained=pretrained, **model_args)
    return model
```

! авторы ConvNeXt не составили карту модели (т.е. описание модели, набор данных на котором проходило обучение, параметры обучения результаты оценки)

ConvNeXT

- Сверточная модель (ConvNet), вдохновленная дизайном Vision Transformers. Авторы начали с ResNet и «модернизировали» его дизайн, взяв за основу Swin Transformer.

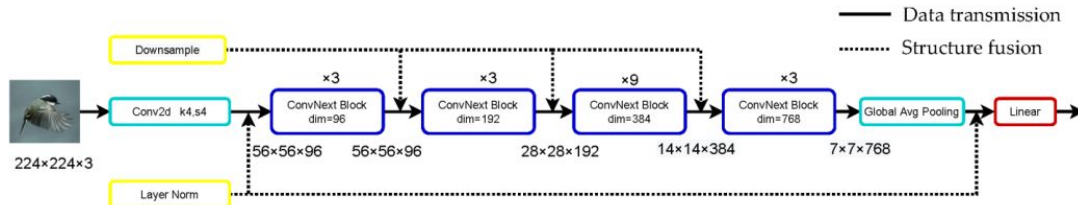
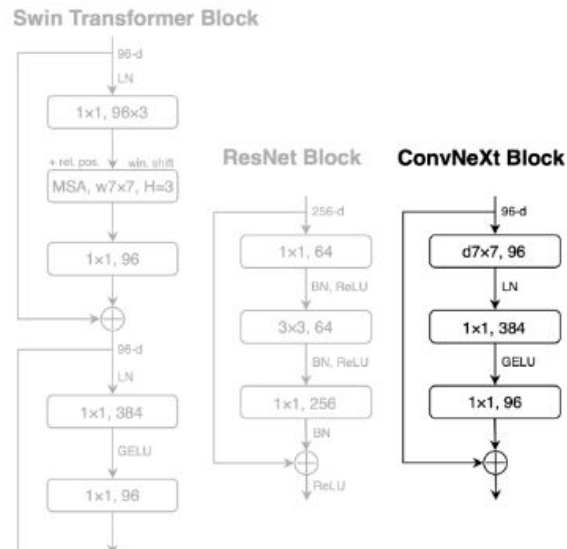
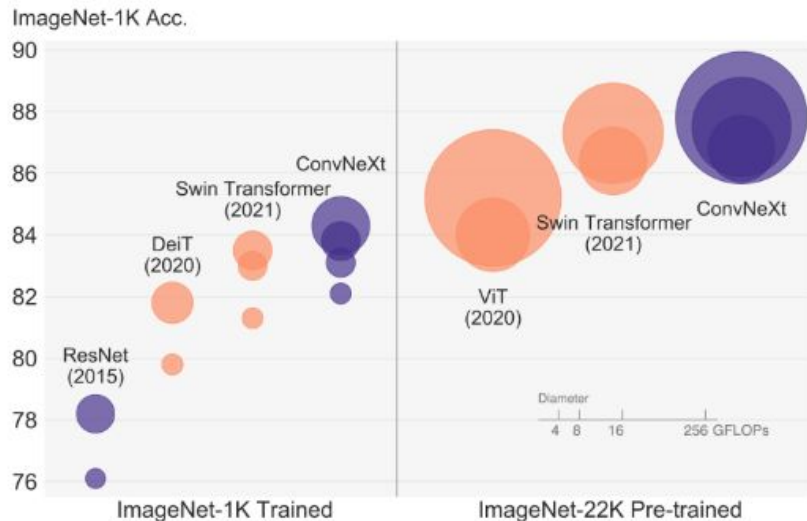


Figure 3. Network structure chart of ConvNeXt-Tiny.

Наборы данных:

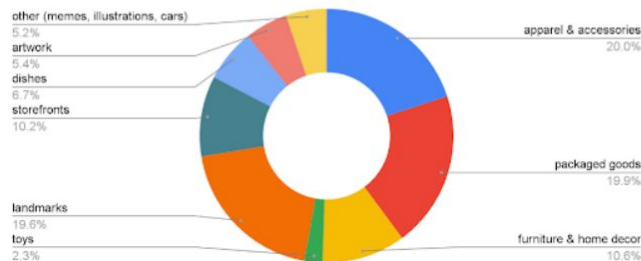
Dataset Description

In this competition, you are asked to develop models that can efficiently retrieve images from a large database. For each image, the model is expected to retrieve the most similar images from an index set.

We do not provide a training set. In accordance with the [rules](#), any training data may be used, as long as it is disclosed in forum by the relevant deadline. There exist many public datasets for different object types (e.g., artworks, landmarks, products etc), and we encourage participants to experiment with them as needed.

Our evaluation dataset, which is kept private, contains images of the following types of object: apparel & accessories, packaged goods, landmarks, furniture & home decor, storefronts, dishes, artwork, toys, memes, illustrations and cars. The header image above shows examples of images in these domains, which are similar to the ones in the evaluation set. Labels are defined generally at the instance level, for example: the same T-shirt, the same building, the same painting, the same dish.

The following plot shows the distribution of object types in the dataset:



Imagenet-22k - полная версия датасета

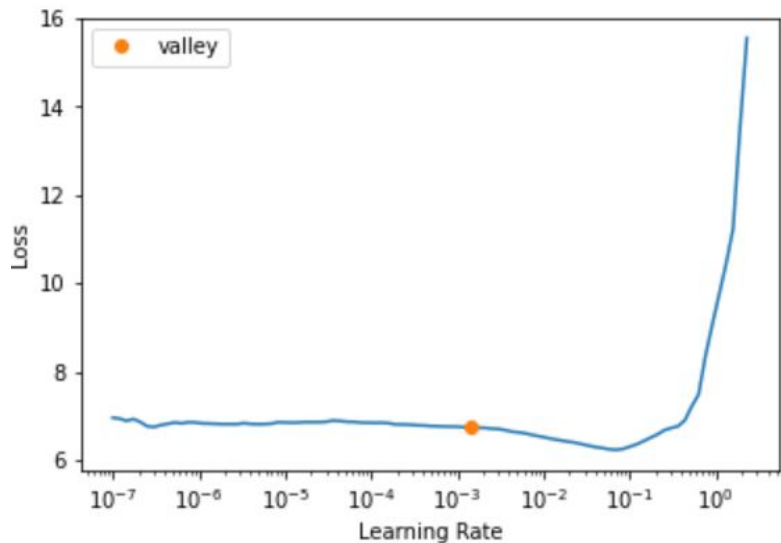
ImageNet -

<https://paperswithcode.com/dataset/imagenet>

Universal Image Embeddings -

<https://www.kaggle.com/competitions/google-universal-image-embedding>

Поиск оптимального learning rate



`learn.lr_find()`

Коэффициент скорости обучения — это параметр градиентных алгоритмов обучения нейронных сетей, позволяющий управлять величиной коррекции весов на каждой итерации.

Более подробно о работе метода:

<https://www.novetta.com/2021/03/learning-rate/>

Метрики: (библиотека sklearn)

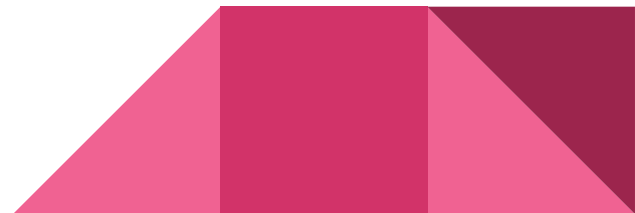
1. точность $Precision = \frac{TP}{TP + FP}$

2. полнота $Recall = \frac{TP}{TP + FN}$

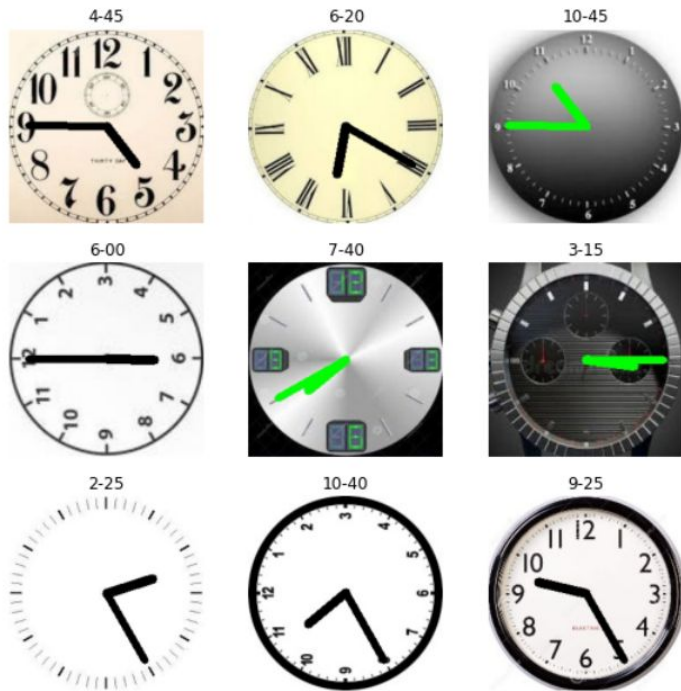
3. f1-мера $F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$

4. accuracy - описывает общую точность предсказания модели по всем классам

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$



Набор данных для обучения:



TIME - Image Dataset-Classification*

- синтетически сгенерированные изображения часов со стрелками
- **144 класса** (формат: час-минута)
- точность - 5 мин
- 224x224x3, .jpg
- поворот на 0, -90, +90 и 180 градусов
- **более 13 тысяч** картинок для обучения

* данные были отредактированы (удалены фотографии часов без стрелок, а также часы с неоднозначным временем)

Поиск оптимальной модели

- Общее время тренировки тестируемых моделей: 36 часов на видеокарте TESLA T4
- Время тренировки лучшей модели: 2 часа
- Лучшая модель: `model_type = 'convnext_xlarge_in22k' epochs = 40`
- Размер обученной модели: 4.2 Гбайт



	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	9
1	1.0000	0.9000	0.9474	10
2	0.9091	1.0000	0.9524	10
3	1.0000	1.0000	1.0000	10
4	0.9000	0.9000	0.9000	10
5	1.0000	1.0000	1.0000	10
...
143	1.0000	0.9000	0.9474	10
accuracy			0.9771	1439
macro avg	0.9793	0.9771	0.9771	1439
weighted avg	0.9793	0.9771	0.9771	1439

Фотографии для проверки:



Проверка производилась на 27 фотографиях часов, сделанных вручную. Собранный датасет включает в себя изображения циферблатов часов разного разрешения и ориентации в пространстве.



верно ↑
почти верно ↓



3-00
3-30

5-50
4-55

9-10
9-15

3-00
12-00

6-55
12-55

9-25
2-25



верны минуты ↑
неверно ↓



	precision	recall	f1-score	support
1	1.0000	1.0000	1.0000	1
6	0.0000	0.0000	0.0000	0
33	1.0000	1.0000	1.0000	1
36	0.0000	0.0000	0.0000	0
46	0.0000	0.0000	0.0000	1
47	0.0000	0.0000	0.0000	0
...
139	1.0000	1.0000	1.0000	1
accuracy			0.4444	27
macro avg	0.3434	0.3333	0.3283	27
weighted avg	0.4938	0.4444	0.4506	27

Для более справедливой оценки работы алгоритма на реальных данных была введена следующая метрика:

Модель получала следующие баллы:

- 0 - если время на картинке определено полностью неверно
- 0.5 - за верное определение времени одной любой стрелки
- 1 - за полностью верное определение времени

Далее вычислялось отношение: $\frac{\sum \text{баллов}}{\text{кол} - \text{во}} \times 100\%$

В результате общая оценка составила: 64.81



Используемая литература:

- <https://www.tocit.ru/static/files/82071b530c60ab12d4f4a1fbdff04dcf87449fd989752855e838684cbf78646b.pdf> - книга по fastai
- <https://medium.com/unpackai/preseizing-techniques-for-image-classification-using-fastai-660a493cfc96> - статья о методах resize, batch_tfms
- <https://docs.fast.ai/vision.augment.html> - fastai documentation
- <https://huggingface.co/facebook/convnext-base-384-22k-1k> - ConvNeXT





Спасибо за внимание!

лист со структурой презентации :

- постановка задачи , предварительная работа
- структура проекта для решения задачи
- библиотеки
- модель сети, описание деталей
- метрики
- датасет для обучения
- результаты работы сети на исходном датасете
- датасет для проверки
- результаты работы сети на нашем датасете
- новые метрики
- литература