

# Movie by Frame

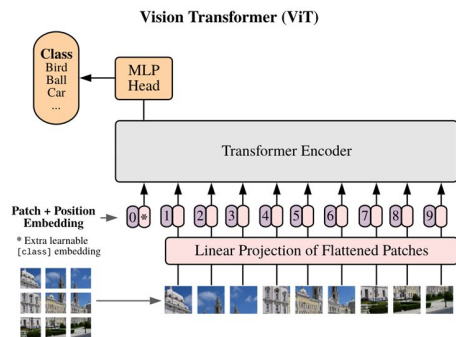
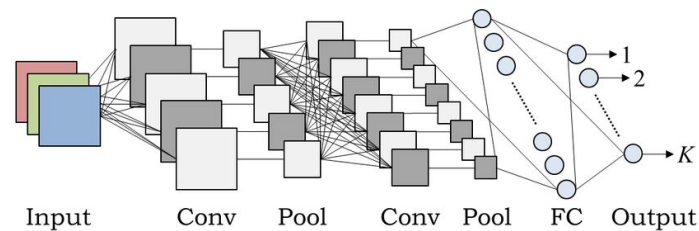
Вдовенков Сергей  
Краснов Алексей  
Мячева Елена

# Основные задачи

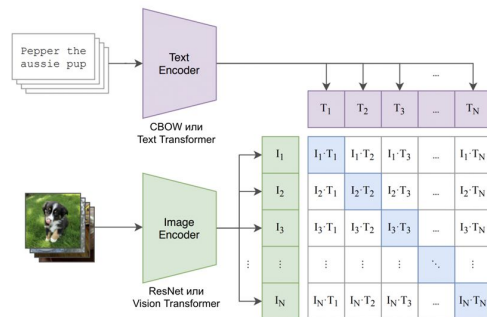
- ❑ Изучение существующих способов решения задачи
- ❑ Поиск и подготовка датасета
- ❑ Реализация моделей
- ❑ Анализ обученных моделей
- ❑ Разработка телеграм-бота



# Способы решения задачи



CNN

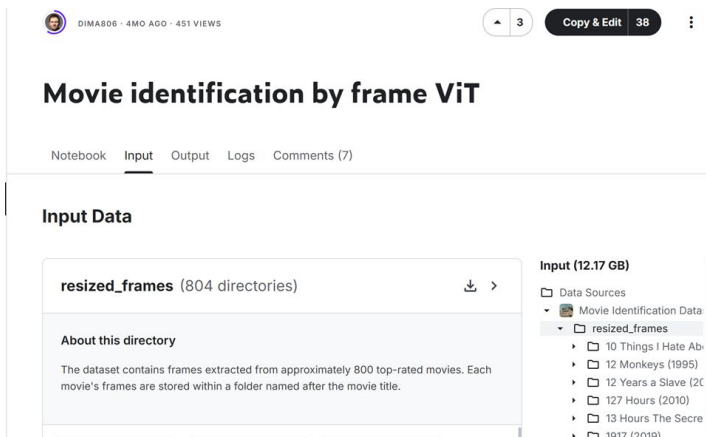


CLIP

ViT

# Поиск датасета

Kaggle



The screenshot shows a Kaggle dataset page. At the top, there's a header with the user profile 'DIMAS06', '4MO AGO', '451 VIEWS', and a 'Copy & Edit' button. Below this is the title 'Movie identification by frame ViT' and tabs for 'Notebook', 'Input', 'Output', 'Logs', and 'Comments (7)'. The 'Input Data' section is active, showing a directory named 'resized\_frames' with 804 directories. A description states: 'The dataset contains frames extracted from approximately 800 top-rated movies. Each movie's frames are stored within a folder named after the movie title.' On the right, a file explorer shows the directory structure: 'Data Sources' > 'Movie Identification Data' > 'resized\_frames' > '10 Things I Hate About You (1999)', '12 Monkeys (1995)', '12 Years a Slave (2013)', '127 Hours (2010)', '13 Hours: The Secret Siege (2016)', and '1917 (2019)'.

IMDb

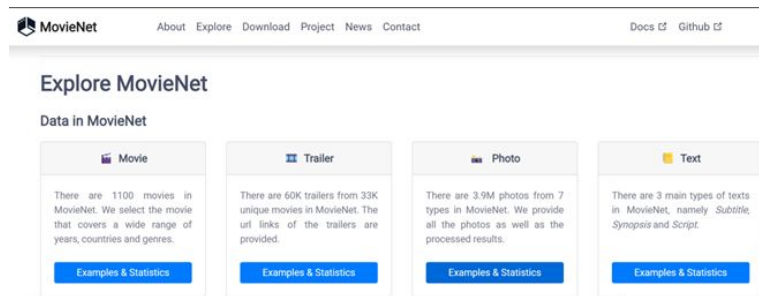


The header of the IMDb Developer page, featuring the IMDb logo, the word 'Developer', and links for 'Documentation' and 'Contact Us'.

## IMDb Non-Commercial Datasets

Subsets of IMDb data are available for access to customers for personal and non-commercial use. You can hold local copies of this data, and it is subject to our terms and conditions. Please refer to the [Non-Commercial Licensing](#) and [copyright/license](#) and verify compliance.

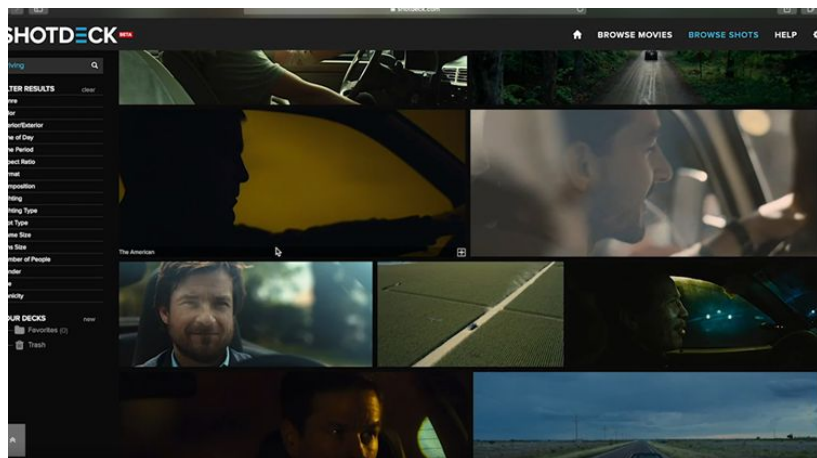
MovieNet



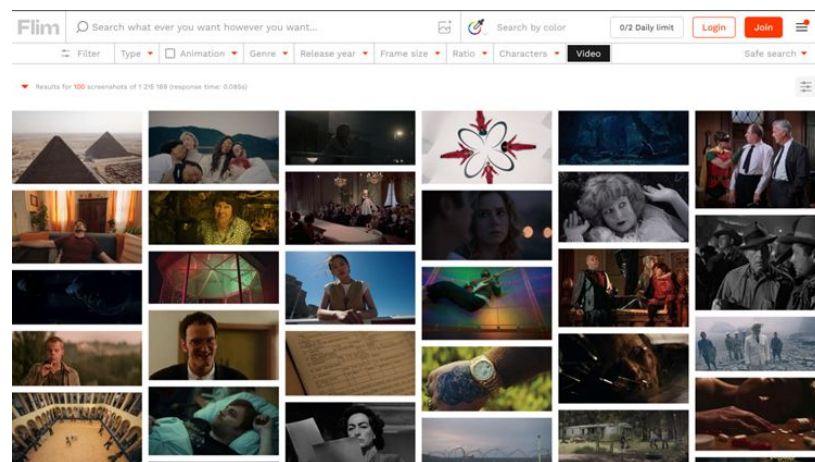
The screenshot shows the MovieNet website. The header includes the MovieNet logo, navigation links (About, Explore, Download, Project, News, Contact), and links for 'Docs' and 'Github'. The main section is titled 'Explore MovieNet' and 'Data in MovieNet'. It features four cards: 'Movie' (1100 movies), 'Trailer' (60K trailers), 'Photo' (3.9M photos), and 'Text' (3 main types of texts: Subtitle, Synopsis, and Script). Each card has an 'Examples & Statistics' button.

# Поиск датасета

Shotdeck



Flim



# Поиск датасета

В итоге был получен датасет с **450+** фильмами.

## Наш датасет

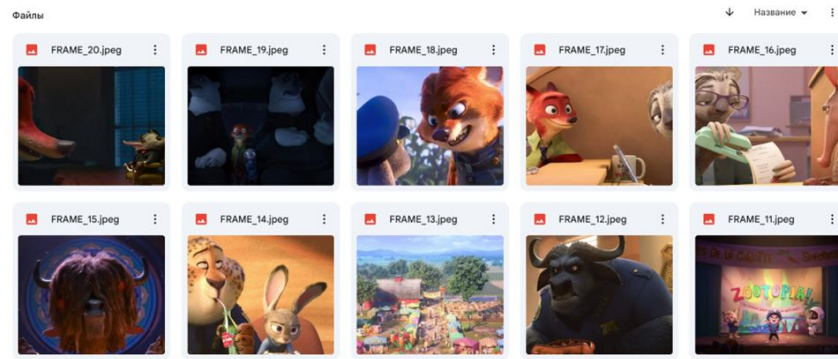
Доступные мне > dataset

Тип Люди Изменено Источник

Название ↑	Послед... ▼
10 Things I Hate About You (1999)	26 янв. 2025 г.
12 Years a Slave (2013)	26 янв. 2025 г.
127 Hours (2010)	26 янв. 2025 г.
300 (2006)	26 янв. 2025 г.
500 Days Of Summer (2009)	26 янв. 2025 г.

Папки

Zootopia (2016)	Zodiac (2007)	Zero Dark Thirty (2012)
X-Men (2000)	Wrath Of Man (2021)	World War Z (2013)
Wonder (2017)	Wind River (2017)	Willow (1988)



# Парсинг датасета

## Получение id фильма

```
curl 'https://api.flim.ai/2.0.0/suggest-entities' \
```

```
-H 'accept: application/json' \
```

```
-H 'content-type: application/json' \
```

```
--data-raw '{"media_type":"","suggest":"joker"}'
```

## Получение кадров

```
curl 'https://api.flim.ai/2.0.0/search' \
```

```
-H 'accept: application/json' \
```

```
-H 'content-type: application/json' \
```

```
--data-raw '{  
  "search": {  
    "movie_id": "%s",  
    ...  
  },  
  "page": 0,  
  "number_per_pages": 100  
}'
```

The screenshot shows a web browser window with the FLIM API interface. The interface includes a search bar, a 'Login' button, and a 'Join' button. Below the search bar, there are two movie posters: one of a man in a red suit (Joker) and another of a man in a white shirt and yellow vest (Joker). To the right of the browser window, a network tool (Fiddler) is open, showing the 'Network' tab. The tool displays a list of network requests, including 'collect', 'statistics', 'suggest-entities', and 'search'. The 'search' request is selected, and its response is shown in the 'Preview' tab. The response is a JSON object containing the following data:

```
{  
  "query_response": {  
    "number_of_results": 100, "total_number_of_result": 1244036,  
    "available_filters": {  
      "genres": [{  
        "value": "DRAMA",  
        "number_of_hits": 100,  
        "total_number_of_results": 1244036  
      }]  
    },  
    "query_response": {  
      "number_of_results": 100, "total_number_of_result": 1244036,  
      "images": [{  
        "id": "c0b5141f-8a30-442c-8f7d-2f8391f8f3d5",  
        "category": "JOKER",  
        "internal_delay": 105,  
        "number_of_results": 100,  
        "total_number_of_results": 1244036  
      }]  
    }  
  }  
}
```

FLIM API

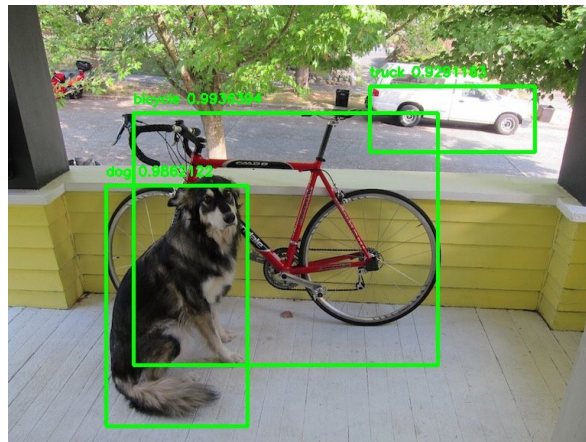
# Подготовка датасета

YOLO – современный алгоритм глубинного обучения, который широко используется для детектирования объектов.

```
def has_person(image_path, model):
    try:
        results = model(image_path, verbose=False)
    except Exception as e:
        print(f"Error processing image {image_path}: {e}")
        return False

    for r in results:
        for obj in r.bboxes.cls:
            # 0 - class `person` from YOLO classes
            if int(obj) == 0: return True
    return False
```

В итоге осталось **250+** фильмов



```
# 1) Dataset with 5 random movies
random_movies_5 = random.sample(processed_movies, 5)
create_dataset_archive("movies_5_processed", random_movies_5, "processed_dataset")

# 2) Dataset with 50 random movies
random_movies_50 = random.sample(processed_movies, 50)
create_dataset_archive("movies_50_processed", random_movies_50, "processed_dataset")

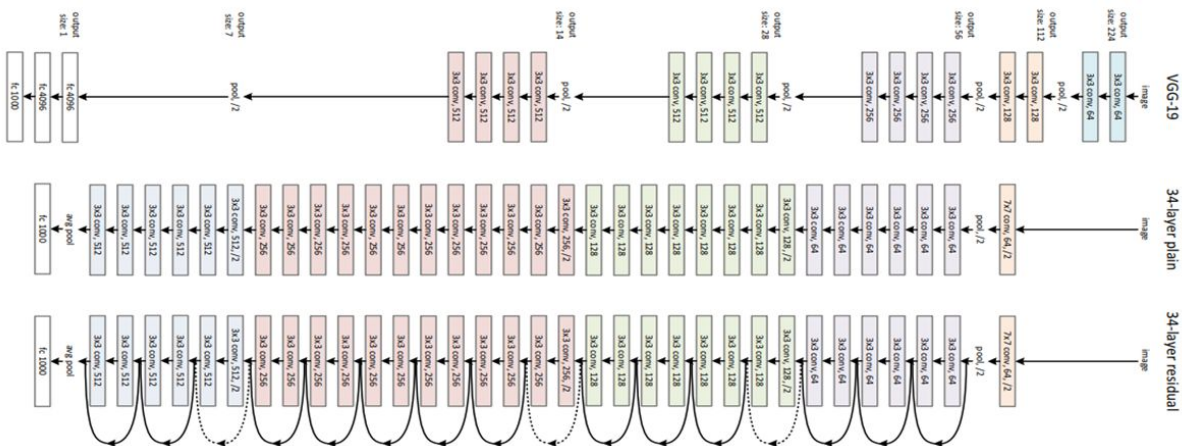
# 3) Dataset with all movies
create_dataset_archive(f"movies_all_processed", processed_movies, "processed_dataset")
```



# CNN

**ResNet** (Residual Network) – это архитектура свёрточной нейронной сети (CNN), разработанная Microsoft в 2015 году.

Основная идея – добавление “быстрых соединений” между слоями. Они позволяют градиентам течь напрямую от выхода одного слоя к входу другого, обходя промежуточные слои.

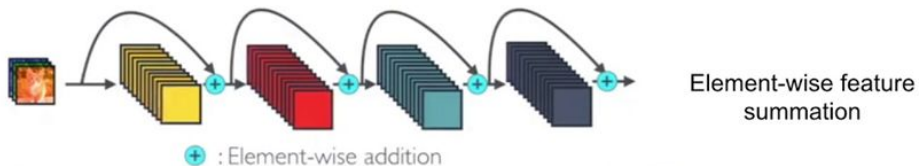


# CNN

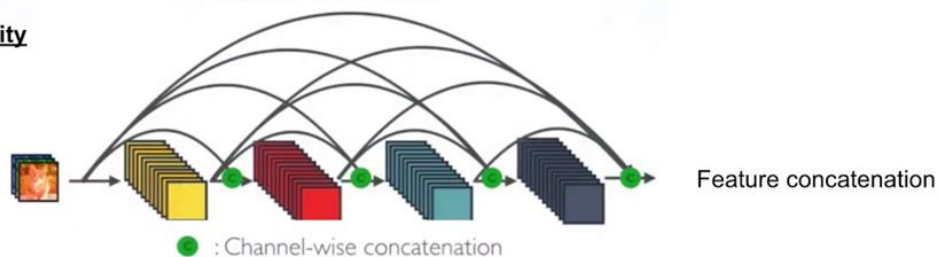
**DenseNet** (Densely Connected Convolutional Network) - это тип свёрточной нейронной сети, разработанной учёными из Корнеллского университета в 2017 году.

Основная идея – соединение каждого слоя со всеми предыдущими, что улучшает обучение за счёт повторного использования признаков.

## Resnet Connectivity



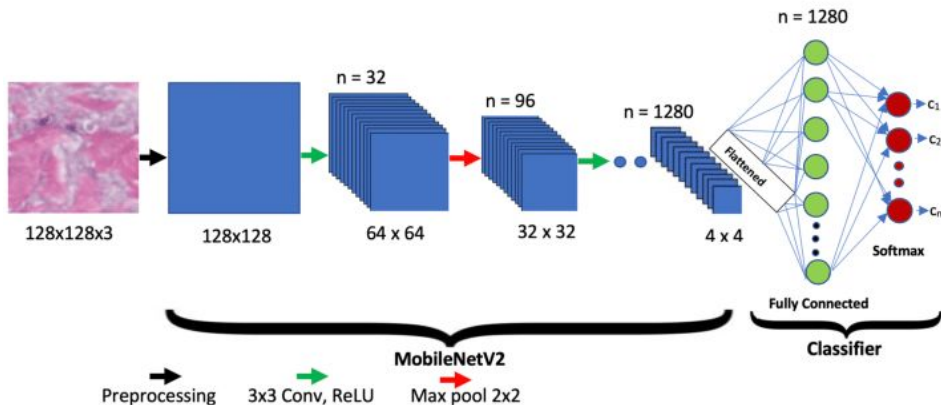
## DenseNet Connectivity



# CNN

**MobileNetV2** – это лёгкая и эффективная архитектура нейронной сети, разработанная для задач компьютерного зрения, таких как классификация изображений, детекция объектов и сегментация.

Основная идея – использование комбинации различных свёрток и архитектурных решений, направленных на достижение высокой производительности при низких вычислительных затратах.



# Дообучение CNN

```
def train_cnn(model_arch, data_path, epochs=10):
    seed_everything(13)

    train_loader, val_loader = dataloaders[data_path]["train"], dataloaders[data_path]["val"]
    num_classes = len(datasets[data_path]["val"].classes)

    run_name = f"{model_arch}-{Path(data_path).stem.replace('_split', '')}".replace("_", "-")
    logger = WandbLogger(project="movie-by-frame", name=run_name, log_model='all')

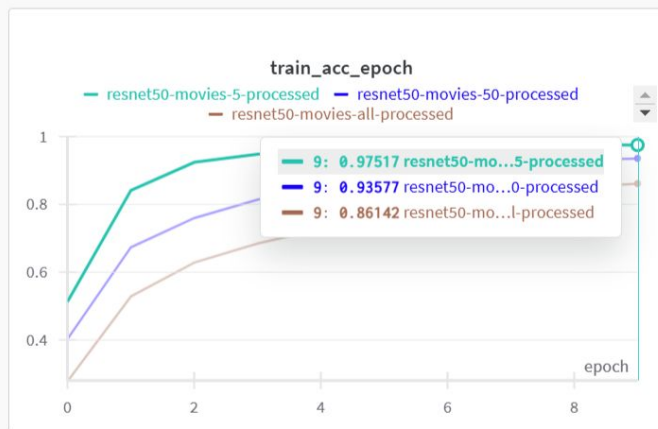
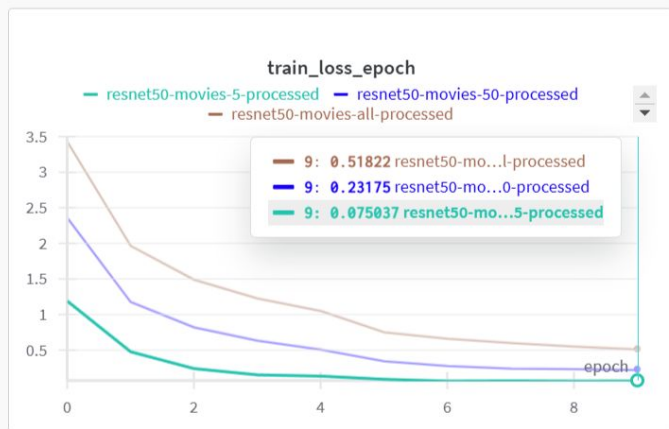
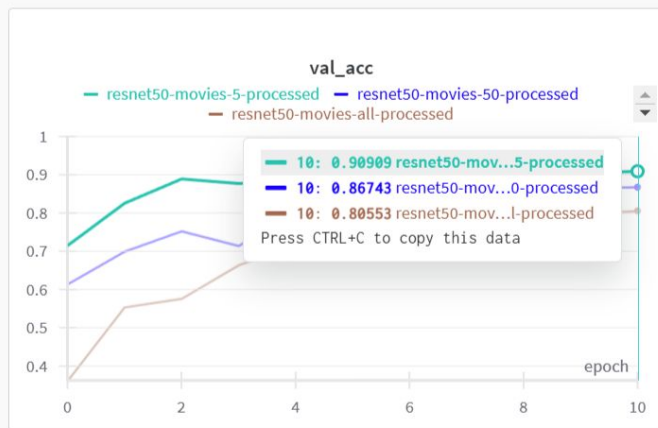
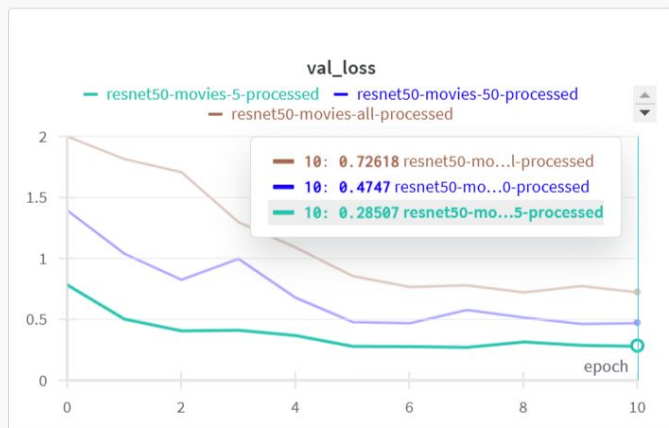
    model = CNNClassifier(model_name=model_arch, num_classes=num_classes)
    checkpoint = ModelCheckpoint(monitor="val_acc", mode="max", save_top_k=1,
                                filename=f"{run_name}-best", save_last=True)

    trainer = pl.Trainer(max_epochs=epochs, accelerator=device, logger=logger, callbacks=[checkpoint],
                          enable_model_summary=False, log_every_n_steps=15)
    trainer.fit(model, train_loader, val_loader)

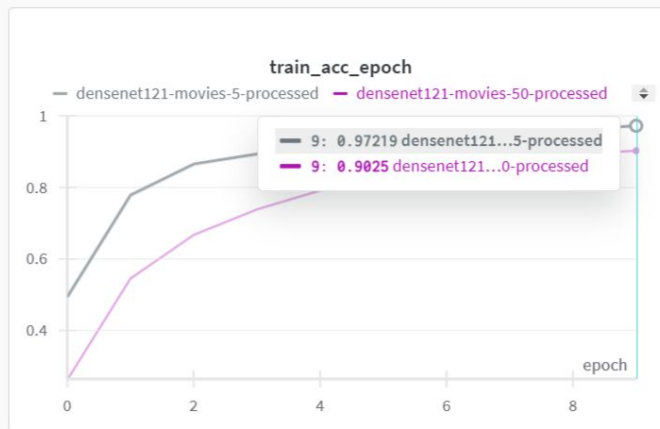
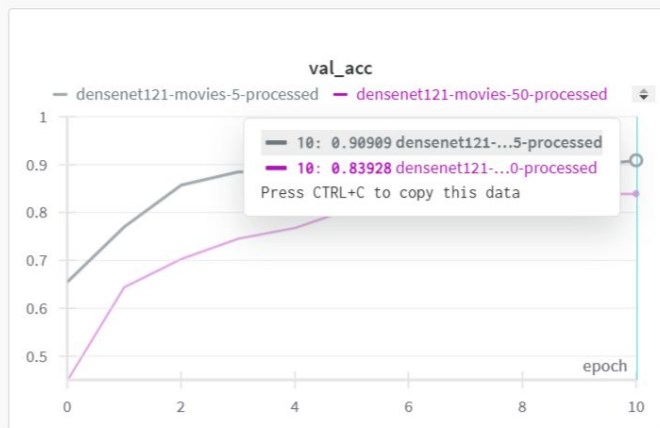
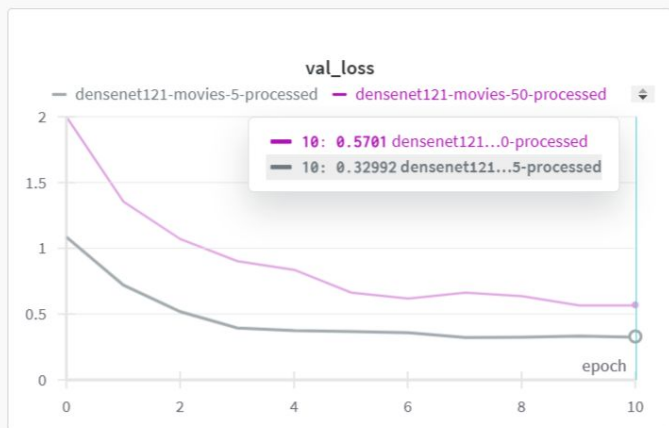
    best_model = CNNClassifier.load_from_checkpoint(checkpoint.best_model_path)
    torch.save(best_model.state_dict(), f"/content/drive/MyDrive/weights/{run_name}-best-weights.pt")

    print(trainer.validate(best_model, val_loader))
    wandb.finish()
```

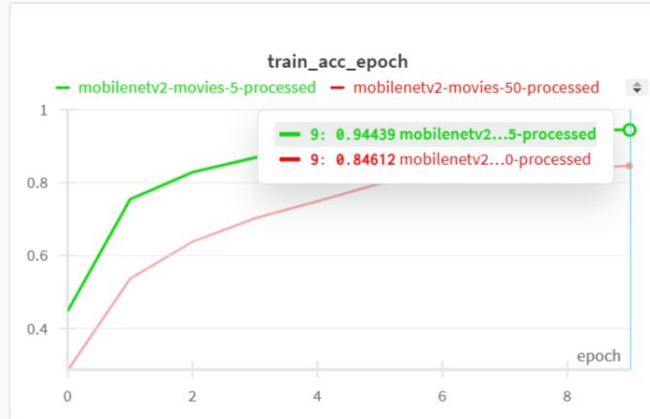
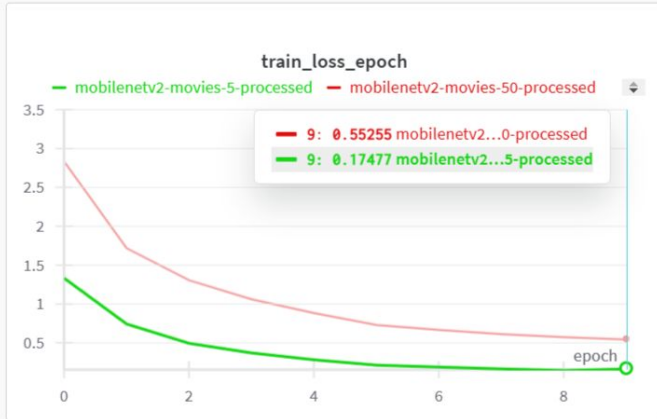
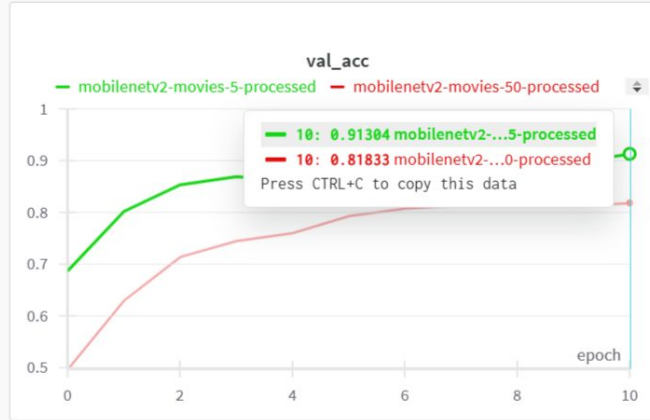
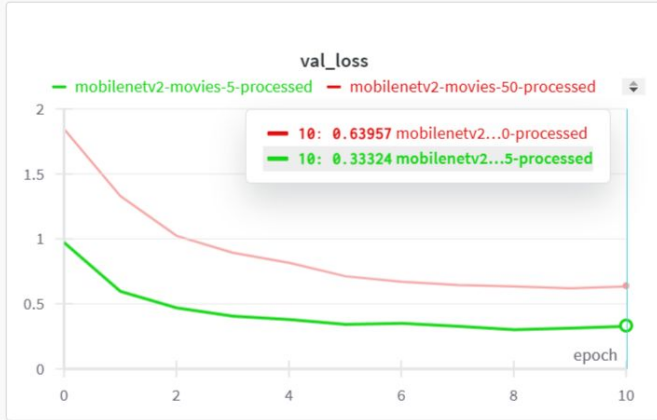
# Resnet



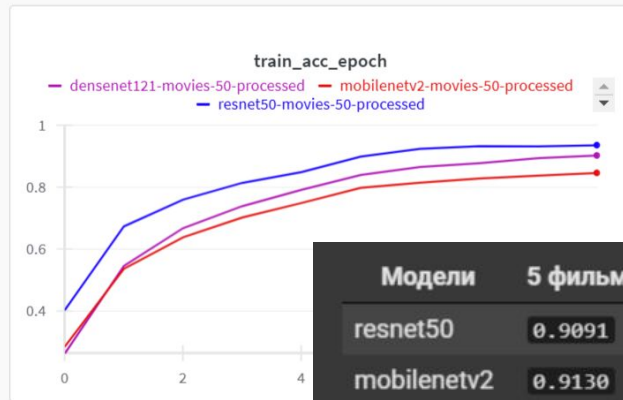
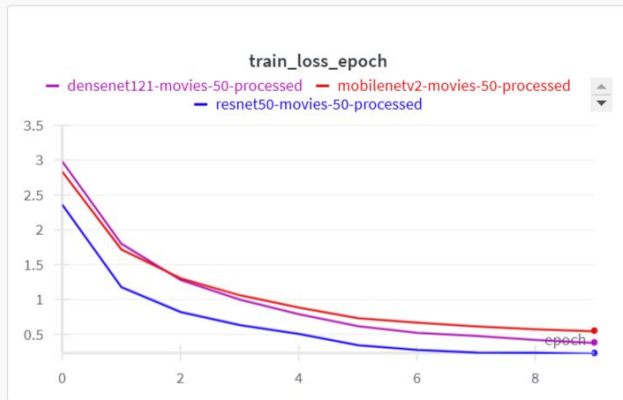
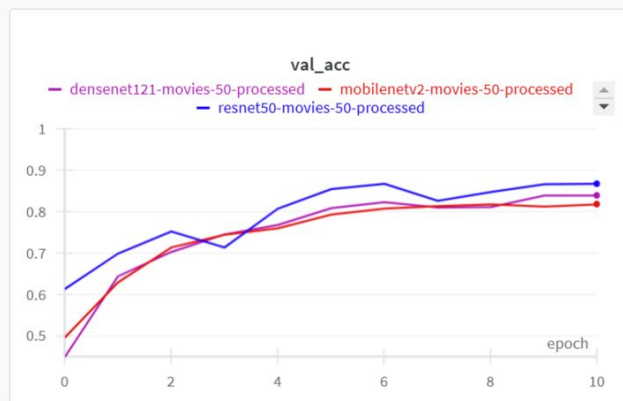
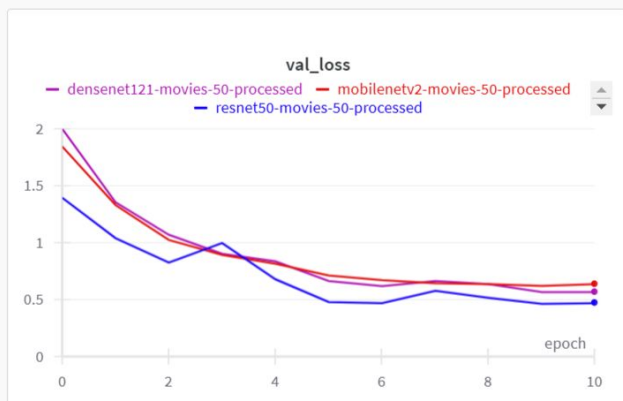
# DenseNet



# MobileNet



# Сравнение



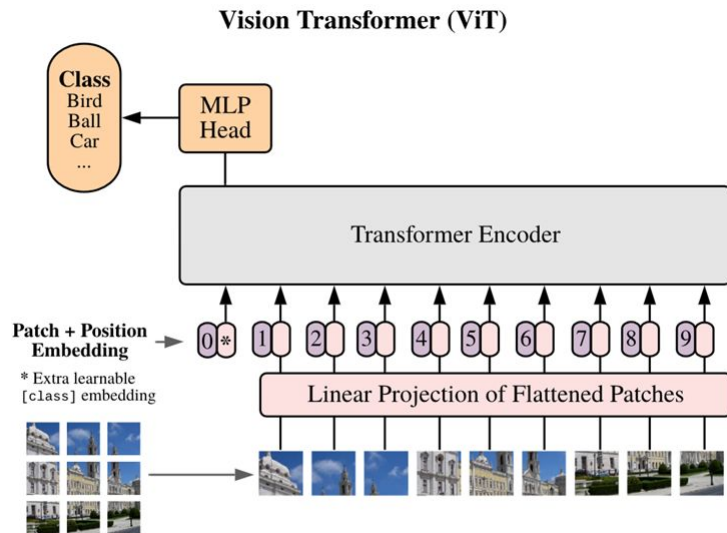
Модели	5 фильмов	50 фильмов	250+ фильмов
resnet50	0.9091	0.8674	0.8055
mobilenetv2	0.9130	0.8183	-
densenet121	0.9091	0.8398	-



# ViT

**ViT (Vision Transformer)** — это архитектура нейросети, предложенная исследователями из Google Research в 2020 году. Это первая модель, использующая **трансформеры** для **анализа изображений**.

Основная идея — работа с изображением как с последовательностью патчей (фрагментов) и использование механизма **self-attention** для выявления глобальных зависимостей.



# DeiT

**DeiT (Data-efficient Image Transformer)** – улучшенная версия ViT от Facebook (Meta), которая быстрее и эффективнее обучается на небольшом наборе данных. Более того, эта модель показывает хорошее качество даже без предобучения.

The screenshot shows the Hugging Face interface for the model `facebook/deit-base-patch16-224`. At the top, it displays the repository name, a 'like' button with 13 likes, a 'Follow' button, and the repository owner 'AI at Meta' with 5.19k followers. Below this, there are tags for 'Image Classification', 'Transformers', 'PyTorch', 'TensorFlow', 'imagenet-1k', 'vit', 'Inference Endpoints', and two arXiv links: 'arxiv:2012.12877' and 'arxiv:2006.03677'. The license is listed as 'License: apache-2.0'. The main navigation bar includes 'Model card', 'Files and versions', and 'Community'. On the right side of the navigation bar, there are buttons for 'Train', 'Deploy', and 'Use this model'. The main content area is divided into two columns. The left column contains the title 'Data-efficient Image Transformer (base-sized model)' and a detailed description: 'Data-efficient Image Transformer (DeiT) model pre-trained and fine-tuned on ImageNet-1k (1 million images, 1,000 classes) at resolution 224x224. It was first introduced in the paper [Training data-efficient image transformers & distillation through attention](#) by Touvron et al. and first released in [this repository](#). However, the weights were converted from the [timm repository](#) by Ross Wightman.' Below the description is a disclaimer: 'Disclaimer: The team releasing DeiT did not write a model card for this model so this model card has been written by the Hugging Face team.' The right column shows the download statistics: 'Downloads last month 143,445' with a line graph showing a sharp peak. Below this is the 'Inference Providers' section, which includes a 'NEW' badge and the 'HF Inference API' provider. There is a section for 'Image Classification' with a dashed box containing the text 'Drag image file here or click to browse from your device'. At the bottom right, there are links for '</> View Code' and a 'Maximize' button.

facebook/**deit-base-patch16-224** like 13 Follow AI at Meta 5.19k

Image Classification Transformers PyTorch TensorFlow imagenet-1k vit Inference Endpoints arxiv:2012.12877 arxiv:2006.03677

License: apache-2.0

Model card Files and versions Community

**Data-efficient Image Transformer (base-sized model)**

Data-efficient Image Transformer (DeiT) model pre-trained and fine-tuned on ImageNet-1k (1 million images, 1,000 classes) at resolution 224x224. It was first introduced in the paper [Training data-efficient image transformers & distillation through attention](#) by Touvron et al. and first released in [this repository](#). However, the weights were converted from the [timm repository](#) by Ross Wightman.

Disclaimer: The team releasing DeiT did not write a model card for this model so this model card has been written by the Hugging Face team.

Downloads last month  
**143,445**

**Inference Providers** NEW HF Inference API

Image Classification

Drag image file here or click to browse from your device

</> View Code Maximize

# Дообучение ViT

```
def train_model(model_size, dataset_path):
    seed_everything(13)
    data = dataloaders[dataset_path]
    print(f"Training {model_size} on dataset: {dataset_path}")

    model_pref = "-".join(model_size.split("_")[:2])
    dataset_suff = dataset_path.split("/")[-1].replace("_split", "")
    wandb_run_name = f"{model_pref}-{dataset_suff}.replace('_', '-')
    wandb_logger = WandbLogger(log_model='all', project="movie-by-frame", name=wandb_run_name)

    num_classes = len(datasets[dataset_path]["val"].classes)
    model = ViTClassifier(model_name=model_size, num_classes=num_classes)

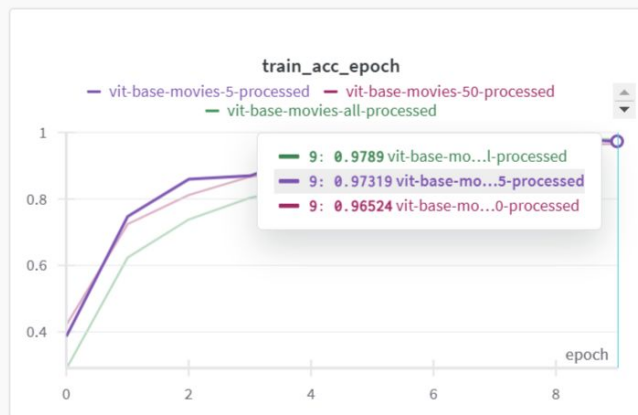
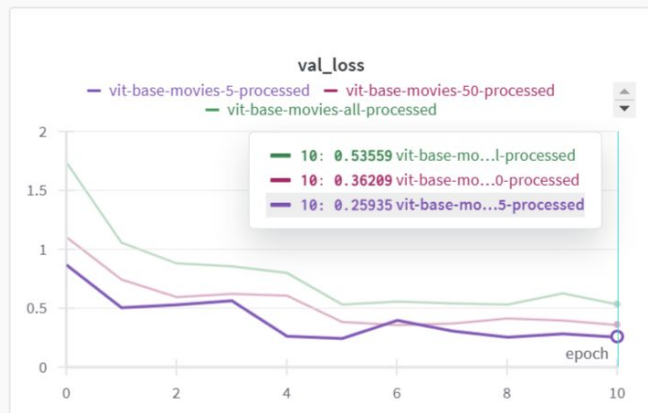
    checkpoint_callback = ModelCheckpoint(
        monitor="val_acc", mode="max",
        save_top_k=1, filename=f"{wandb_run_name}-best",
        save_last=True
    )

    trainer = pl.Trainer(max_epochs=epochs, accelerator=device,
                        logger=wandb_logger, enable_model_summary=False,
                        callbacks=[checkpoint_callback], log_every_n_steps=15)
    trainer.fit(model, data["train"], data["val"])

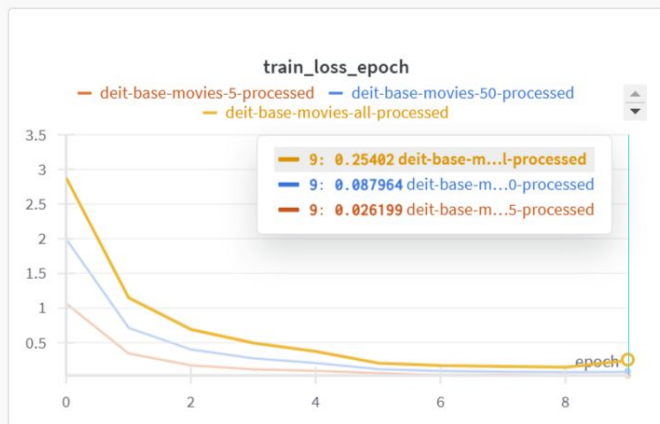
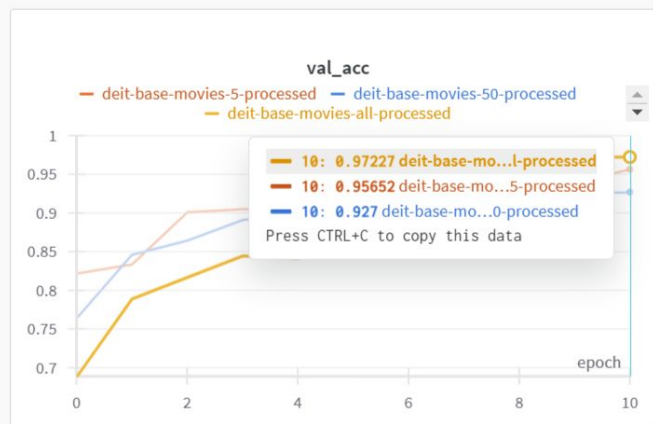
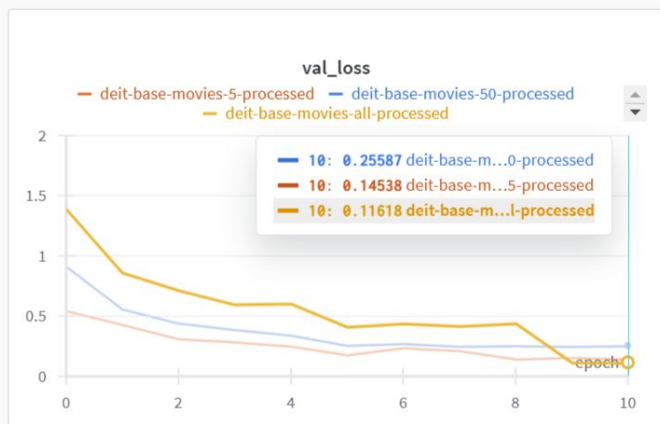
    best_model_path = checkpoint_callback.best_model_path
    best_model = ViTClassifier.load_from_checkpoint(best_model_path)

    val_result = trainer.validate(best_model, data["val"])
    print(val_result)
    model_weights_path = f"/content/drive/MyDrive/weights/{wandb_run_name}-best-weights.pt"
    torch.save(best_model.state_dict(), model_weights_path)

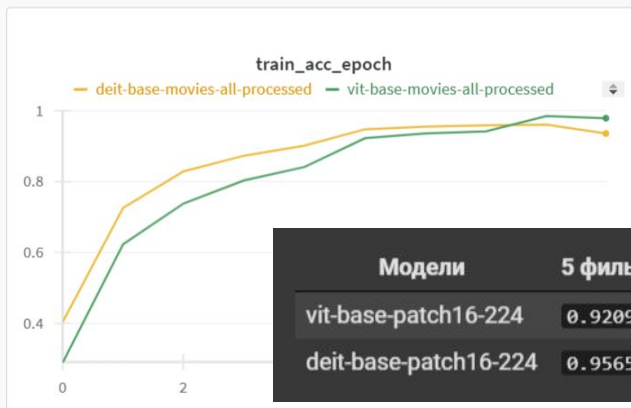
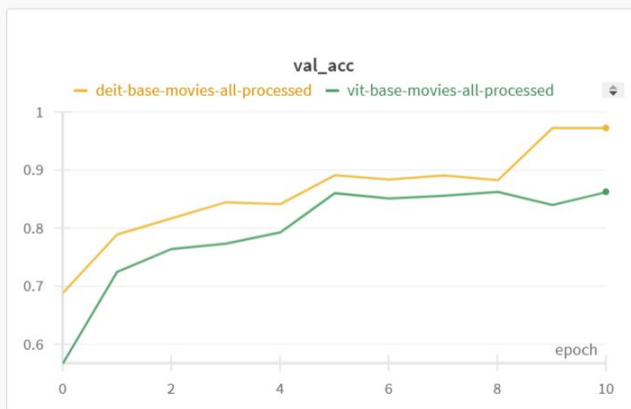
    wandb.finish()
```



# DeiT



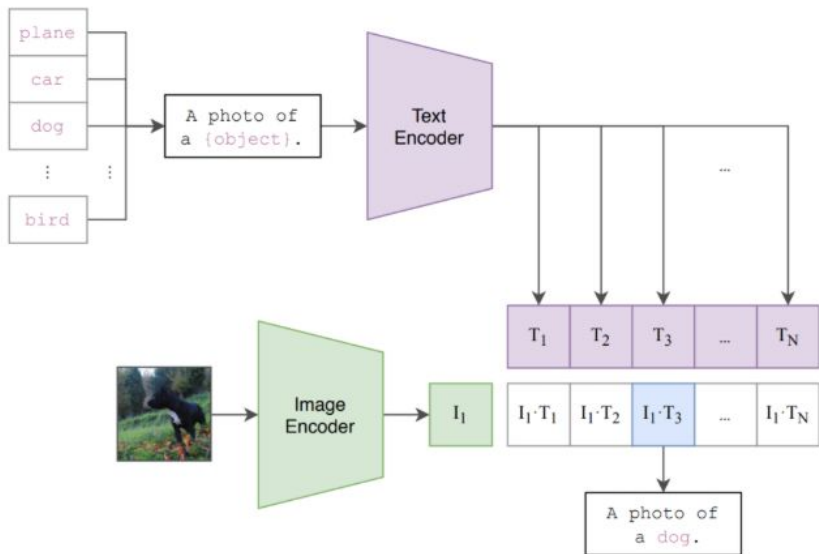
# Сравнение



Модели	5 фильмов	50 фильмов	250+ фильмов
vit-base-patch16-224	0.9209	0.8962	0.8625
deit-base-patch16-224	0.9565	0.9270	0.9722

# CLIP

**CLIP (Contrastive Language-Image Pretraining)** — это модель от OpenAI, которая связывает текст и изображения. Она обучена на огромном количестве изображений с подписями и умеет понимать, какие тексты соответствуют каким картинкам.

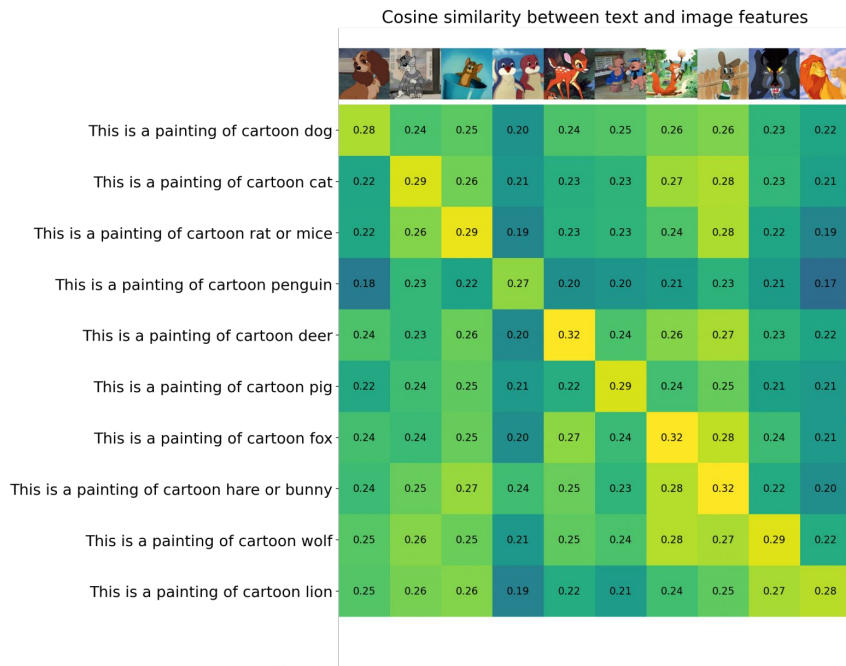


*CLIP использует два нейросетевых энкодера: визуальный энкодер для изображений и текстовый энкодер для текстов.*

# CLIP

Основная идея – контрастивное обучение:

- ❑ Модель получает пары "изображение – текст".
- ❑ Учится сближать представления (векторные представления) правильных пар и отдалять несвязанные.





# Дообучение CLIP

```
def train_model(run_name, model_name, epochs=5):
    seed_everything(13)

    processor = CLIPProcessor.from_pretrained(f"openai/{model_name}")
    train_dataset = MovieDataset(TRAIN_DIR, processor)
    val_dataset = MovieDataset(VAL_DIR, processor)
    train_dataloader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)
    val_dataloader = DataLoader(val_dataset, batch_size=BATCH_SIZE, shuffle=False)

    model = CLIPFineTuner(model_name=model_name)

    wandb_logger = WandbLogger(log_model='all', project="movie-by-frame", name=run_name)
    checkpoint_callback = ModelCheckpoint(
        monitor="val_acc", mode="max",
        save_top_k=1, filename=f"{run_name}-best",
        save_last=True
    )

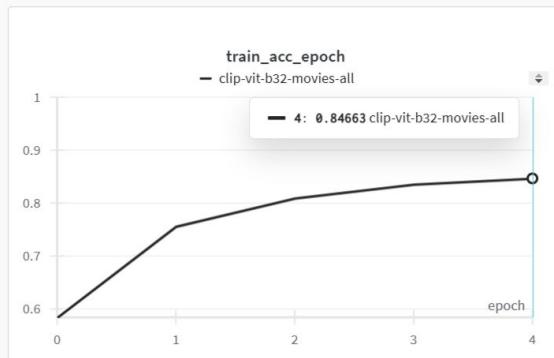
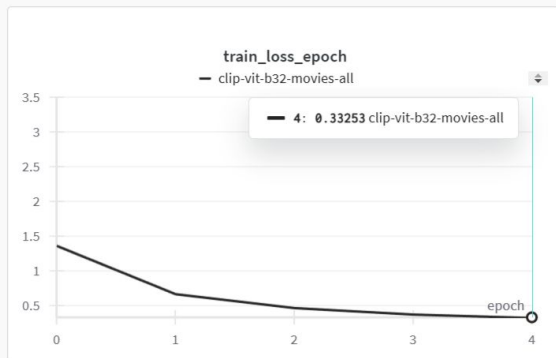
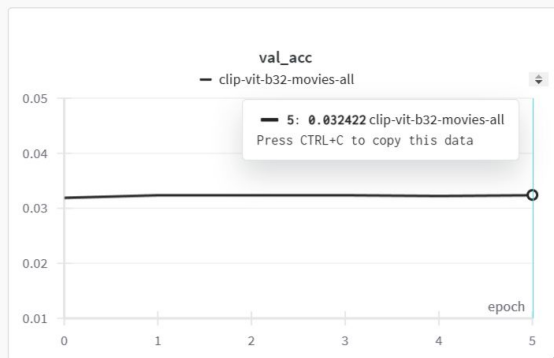
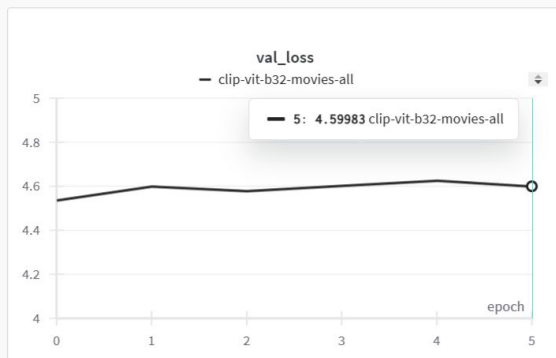
    trainer = pl.Trainer(
        max_epochs=epochs, accelerator=device,
        logger=wandb_logger, enable_model_summary=False,
        callbacks=[checkpoint_callback], log_every_n_steps=15
    )
    trainer.fit(model, train_dataloader, val_dataloader)

    best_model_path = checkpoint_callback.best_model_path
    best_model = CLIPFineTuner.load_from_checkpoint(best_model_path)

    print(trainer.validate(best_model, val_dataloader))
    model_weights_path = f"/content/drive/MyDrive/weights/{run_name}-best-weights.pt"
    torch.save(best_model.state_dict(), model_weights_path)

    wandb.finish()
```

# Результаты CLIP



# Разработка телеграм-бота

```
class ViTClassifier(pl.LightningModule):
    def __init__(self, *, model_name, num_classes):
        super().__init__()
        self.save_hyperparameters()
        self.model = create_model(model_name, pretrained=True, num_classes=num_classes)
        self.model.head = nn.Linear(self.model.head.in_features, num_classes)

    def forward(self, x):
        return self.model(x)

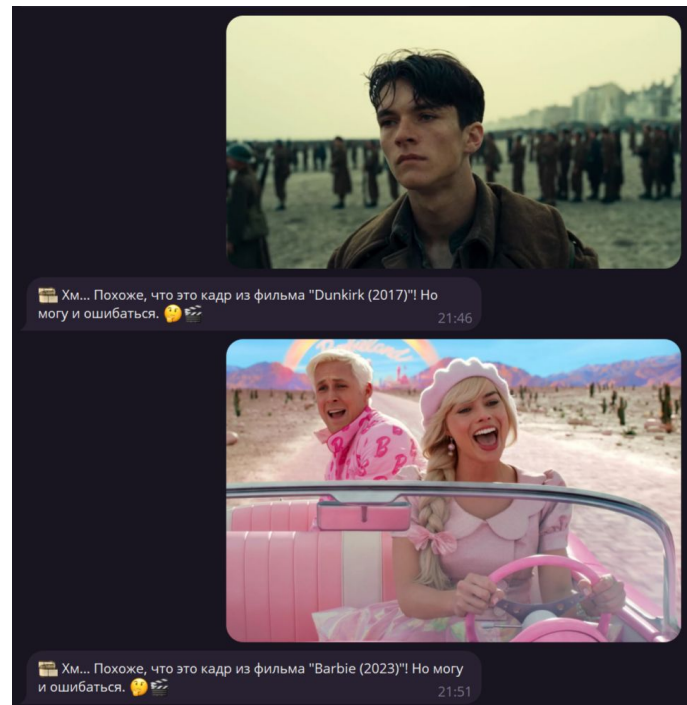
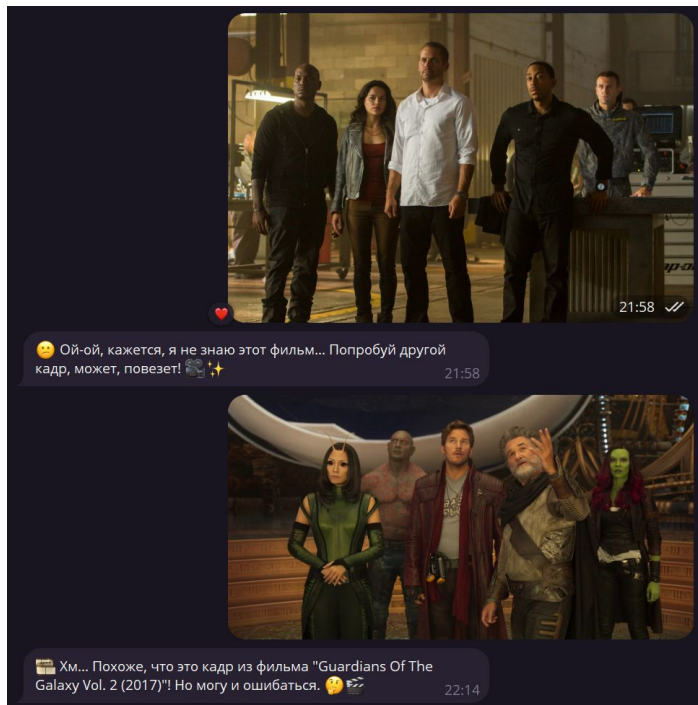
model = ViTClassifier(model_name=MODEL_NAME, num_classes=len(CLASSES))
state_dict = torch.load(MODEL_PT_PATH)
model.load_state_dict(state_dict)
model.eval()
```

```
with torch.no_grad():
    output = model(image_tensor)
    mean = torch.mean(output, dim=1, keepdim=True)
    std = torch.std(output, dim=1, keepdim=True)
    normalized_output = (output - mean) / std
    scaled_output = normalized_output / TEMPERATURE
    probabilities = torch.sigmoid(scaled_output)
```

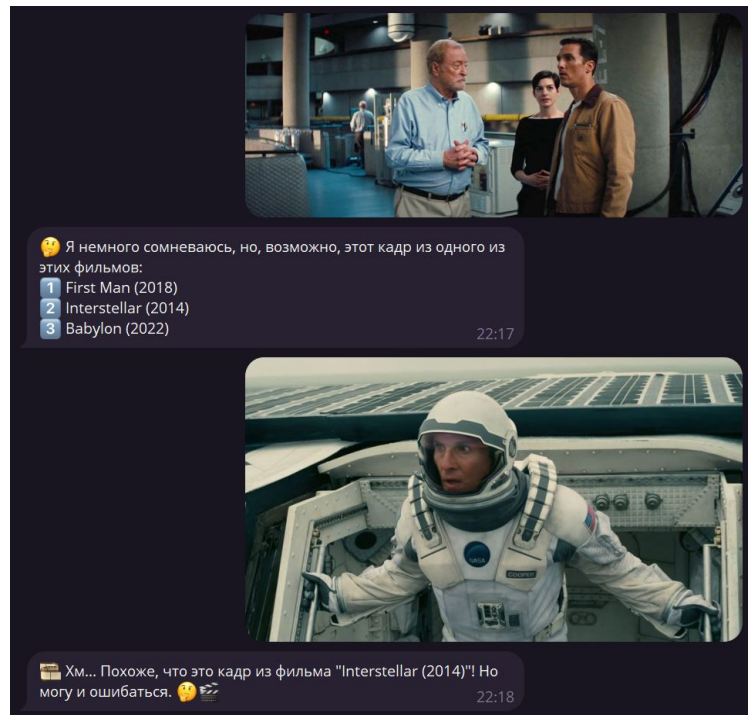
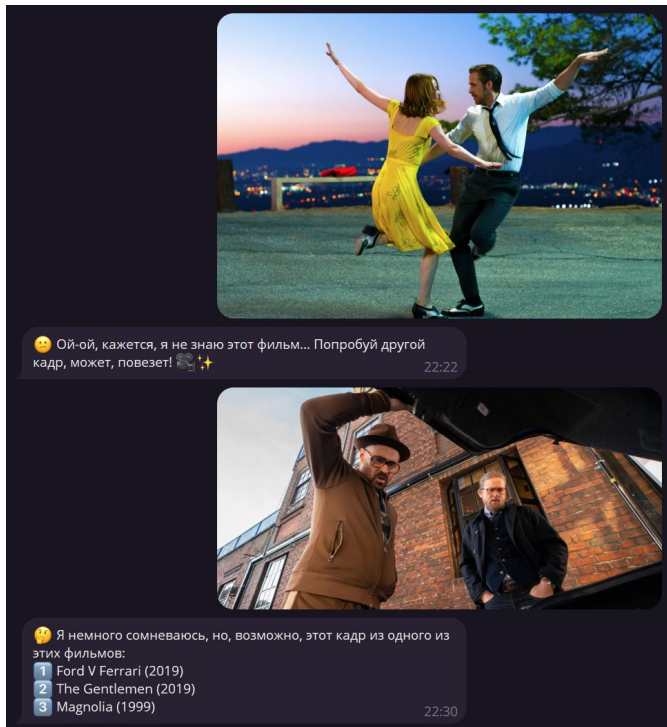
```
MODEL_NAME = "deit_base_patch16_224"
MODEL_PT_PATH = "data/model.pt"
THRESHOLD_TOP1 = 0.9
THRESHOLD_TOP3 = 0.83
TEMPERATURE = 2.5
```

Лучшая модель – deit-base-patch16-224.


# Скрины результатов



# Скрины результатов



# Скрины результатов




21:40 ✓

🧠 Я немного сомневаюсь, но, возможно, этот кадр из одного из этих фильмов:


- 1 Doctor Strange (2016)
- 2 Guardians Of The Galaxy Vol. 2 (2017)
- 3 Doctor Strange In The Multiverse Of Madness (2022)

21:40




📺 Хм... Похоже, что это кадр из фильма "Bridge Of Spies (2015)"! Но могу и ошибаться. 🧠🎬

21:41



📺 Хм... Похоже, что это кадр из фильма "Oblivion (2013)"! Но могу и ошибаться. 🧠🎬

21:52



📺 Хм... Похоже, что это кадр из фильма "Oblivion (2013)"! Но могу и ошибаться. 🧠🎬

21:53



# Выводы



- ❑ **Визуальные трансформеры** показали лучшие результаты благодаря способности анализировать глобальные зависимости в изображениях.
- ❑ **CNN-модели** продемонстрировали среднее качество, так как их локальная обработка ограничивает захват контекста сцены.
- ❑ **CLIP переобучился**, что привело к низкому ассигасу, указывая на необходимость тщательной настройки гиперпараметров и более разнообразного датасета.
- ❑ **Качество данных критично**: ручной отбор кадров улучшает результаты, но увеличивает затраты.
- ❑ **Telegram-бот** стал удобным инструментом для поиска фильмов по кадрам, раскрывая потенциал визуальных трансформеров в реальных задачах.

# Ссылки на github и notebooks

- ❑ <https://github.com/Vdovenkov-Sergei/Movie-by-frame>
- ❑ <https://github.com/UnderPocker/MovieParser>
- ❑ [https://colab.research.google.com/drive/17JxuayjPuozBWO\\_sEiZtAi5U4YshZD8C?usp=sharing](https://colab.research.google.com/drive/17JxuayjPuozBWO_sEiZtAi5U4YshZD8C?usp=sharing)
- ❑ <https://colab.research.google.com/drive/1lpvU-D3UQLfhU6OhUfs0vdBEke9pzWQj?usp=sharing>
- ❑ <https://colab.research.google.com/drive/1YKVuJskrA0oiuiCr7DSQoHd7nOsSvLnU?usp=sharing>
- ❑ <https://colab.research.google.com/drive/159HRAMp3ZIEuRmRsXWFETcSwQldCJU9r?usp=sharing>

