* Видеокурс от Stanford University: <https://www.youtube.com/watch?v=vT1JzLTH4G4&list=PLC1qU-LWwrF64f4QKQT-Vg5Wr4qEE1Zxk>
* Курс от Физтех-школы Прикладной Математики и Информатики МФТИ: <https://vk.com/dlschool_mipt>
* Курс Нейронные сети и компьютерное зрение от Sumsung: <https://stepik.org/course/50352/promo#toc>

[Blur dataset | Kaggle](https://www.kaggle.com/datasets/kwentar/blur-dataset)

<https://colab.research.google.com/drive/1NwMieyhZh9td4KNtUMPKFr0yJ_ROYxCV?usp=sharing>

1. Collect and organize your dataset: Gather images that represent each of the categories you want to classify. It is important to have a sufficient number of images for each class. You can organize your dataset into separate folders for each category.
2. Split your dataset into training, validation, and testing sets: Divide your dataset into three sets: training, validation, and testing. The training set is used to train the model, the validation set is used to tune the hyperparameters and prevent overfitting, and the testing set is used to evaluate the performance of the model.
3. Preprocess your images: Preprocessing your images can help improve the performance of your model. You can resize your images to a consistent size, normalize the pixel values, and apply data augmentation techniques such as random rotation, flipping, and cropping.
4. Encode your labels: For multi-output classification, you will need to encode your labels for **each output separately**. For example, you can use one-hot encoding for the quality output and label encoding for the rotation and RGB-BGR outputs.
5. Load your dataset: Use a framework like TensorFlow or PyTorch to load your dataset into memory. You can use data loaders to efficiently load and preprocess your data in batches.
6. Define your model: Choose an appropriate architecture for your model, such as a CNN, and define the layers and parameters. For multi-output classification, you will need to define multiple output layers with appropriate activation functions.
7. Train and evaluate your model: Train your model using the training set and validate it using the validation set. Monitor the loss and accuracy during training and adjust the hyperparameters as needed. Evaluate the performance of your model on the testing set using appropriate metrics for multi-output classification.
8. Predict with new data: Once your model is trained, you can use it to predict the categories of new images. Preprocess the new images in the same way as your training data and use the model's predict function to generate predictions for each output.

*How to collect the data for text classification problem? I want to gather the data from various sources such as emails, messengers, and website comments. Then preprocess the data by removing stop words, stemming, and lemmatization.*

Data Collection

Collecting data for a text classification problem can be done in a few ways, depending on the specific problem you are trying to solve. Here are some methods:

1. Web scraping: You can scrape data from various sources, such as e-commerce websites, social media platforms, online forums, and blogs. Use web scraping tools like BeautifulSoup and Scrapy to extract text data from web pages.
2. APIs: Some websites provide APIs that allow you to access their data programmatically. For example, you can use the Twitter API to collect tweets or the Reddit API to gather comments from subreddits.
3. Surveys: You can create surveys to collect data from your target audience. Ask questions that relate to your classification problem, and collect the answers as text data.
4. Data brokers: You can purchase pre-collected datasets from data brokers. These datasets may come from various sources, such as social media platforms, customer review websites, and other online sources.

Once you have collected the data, you can preprocess it using techniques such as removing stop words, stemming, and lemmatization. You can use Python libraries like NLTK, spaCy, or scikit-learn to perform these tasks.

It's important to note that the quality of your data will play a significant role in the performance of your text classification model. Make sure to carefully select your sources, and clean and preprocess the data thoroughly before training your model.

**Collecting tweets through Twitter APIs**

Collecting data using the Twitter API is a popular method for gathering text data for text classification tasks. Here are the general steps you can follow to collect and preprocess Twitter data for text classification:

* Create a Twitter Developer Account: First, you need to create a developer account on Twitter and create a new Twitter App. Once you have created an app, you will be provided with access tokens and keys that you can use to make API calls.
* Set Up Your API Request: Use a Python library like Tweepy to make API requests to Twitter. Set up your API request to collect tweets based on specific keywords, user handles, hashtags, or locations.
* Collect Data: Run your API request to collect tweets and store them in a JSON or CSV file.
* Preprocess the Data: Preprocess the collected tweets by removing stop words, stemming, and lemmatization. You can use Python libraries like NLTK or spaCy to perform these tasks.
* Label the Data: Once you have preprocessed the data, label the tweets based on the categories you are interested in.
* Split Data into Training and Test Sets: Split your labeled data into training and test sets. The training set will be used to train your text classification model, and the test set will be used to evaluate its performance.
* Train and Evaluate your Model: Select an appropriate machine learning algorithm, like Naive Bayes, Support Vector Machines, Logistic Regression, Decision Trees, Random Forest, or Neural Networks, and train your model on the labeled data. Evaluate the performance of the model using appropriate metrics like accuracy, precision, recall, and F1 score.
* Deploy your Model: Once you have fine-tuned your model, deploy it to classify new incoming Twitter data based on the categories you have defined.

Remember that the success of your model depends on the quality and quantity of labeled data, as well as the quality of your preprocessing steps. So, make sure to spend enough time on each step to get the best results.