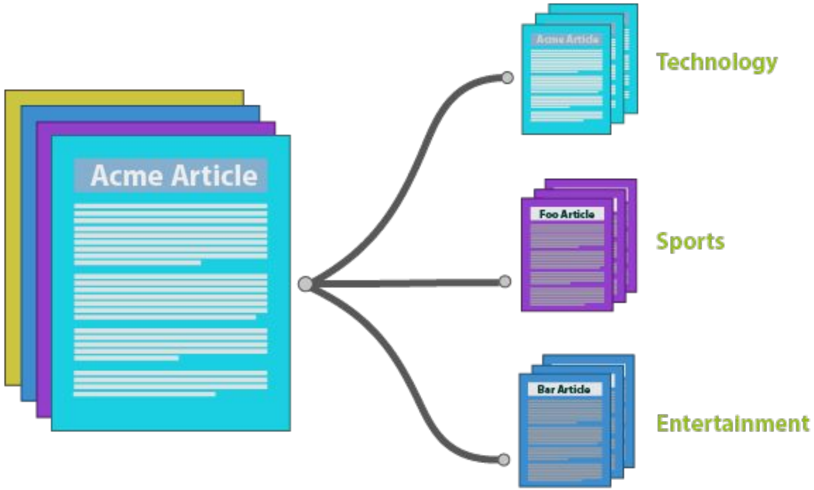
Text classification

- Dataset
- Data Preprocessing
- Transformers for Classification



Dataset

News_train.json

- **Size**: Over **135,000** labeled news samples
- **Format**: Each entry is a JSON object

Fields:

- `id`: Sample index
- `headline`: News headline
- `short_description`: Summary of the news article
- `label`: Numeric category of the article

News_test.json

- **Size**: **1,000** unlabeled news samples
- **Format**: Similarly to `News_train.json`, but without the `label` field.

Label Mapping

- POLITICS → 0
- WELLNESS → 1
- ENTERTAINMENT → 2
- TRAVEL → 3
- STYLE & BEAUTY → 4
- PARENTING → 5
- HEALTHY LIVING → 6
- QUEER VOICES → 7
- FOOD & DRINK → 8
- BUSINESS → 9
- COMEDY → 10
- SPORTS → 11
- BLACK VOICES → 12
- HOME & LIVING → 13
- PARENTS → 14

Text preprocessing: Cleaning

- **Lowercasing**

  Convert all text to lowercase for consistency.

- **Removing Punctuation & Special Characters**

  Eliminate symbols like `. , ! ?@#` that don't add semantic value.

- **Stop-Words Removal**

  Remove common words (e.g., *the, is, in*) that carry little meaning.

"The cats are running quickly!"

↓

"the cats are running quickly!"

↓

"the cats are running quickly"

↓

"cats running quickly"

Text preprocessing: Cleaning

- **Stemming**

  Reduce words to their root form. E.g., "running" → "run"

- **Tokenization**

  Split text into tokens (words or subwords).

- **Token-to-ID (Indexing / Encoding)**

  Map tokens to numerical values by vocabulary ids.

"cats running quickly"

↓

"cats run quickly"

↓

["cats", "run", "quickly"]

↓

[257,14,80]

Text preprocessing: Special Tokens

Special tokens are reserved symbols added to text during preprocessing to help models understand structure, context, and boundaries.

Common Special Tokens	
Token	**Purpose**
<CLS>	Added at the start of input for classification tasks
<SEP>	Separates two segments (e.g., sentence pairs)
<PAD>	Used to pad input sequences to the same length
<MASK>	Used for masked language modeling (e.g., BERT pretraining)
<UNK>	Represents unknown or out-of-vocabulary tokens

Input:

"NLP is amazing!"

After tokenization and special tokens:

<CLS> NLP is amazing ! **<SEP>**

Converted to IDs:

[**101**, **2342**, **2003**, **6429**, **999**, **102**]

Transformers for Classification

🎯 Objective

Implement a **Transformer-based model for text classification**, using **PyTorch or TensorFlow.**

🚫 Restrictions

- ❌ **Do not use high-level libraries such as Hugging Face**
- ✅ **Use built-in Transformer blocks (e.g., encoder layers, positional encoding)**

Transformers for Classification

Build a Transformer consisting of:

Transformer Encoder

Stack multiple layers for deeper models

Positional Encoding

Add information about the position of each token in a sequence

Input Embedding

Token embedding layer

Batch Processing

batching of input sequences

Training Function

Custom train() function and manual training loop

Classification Head

Use [CLS] token or pooling strategy

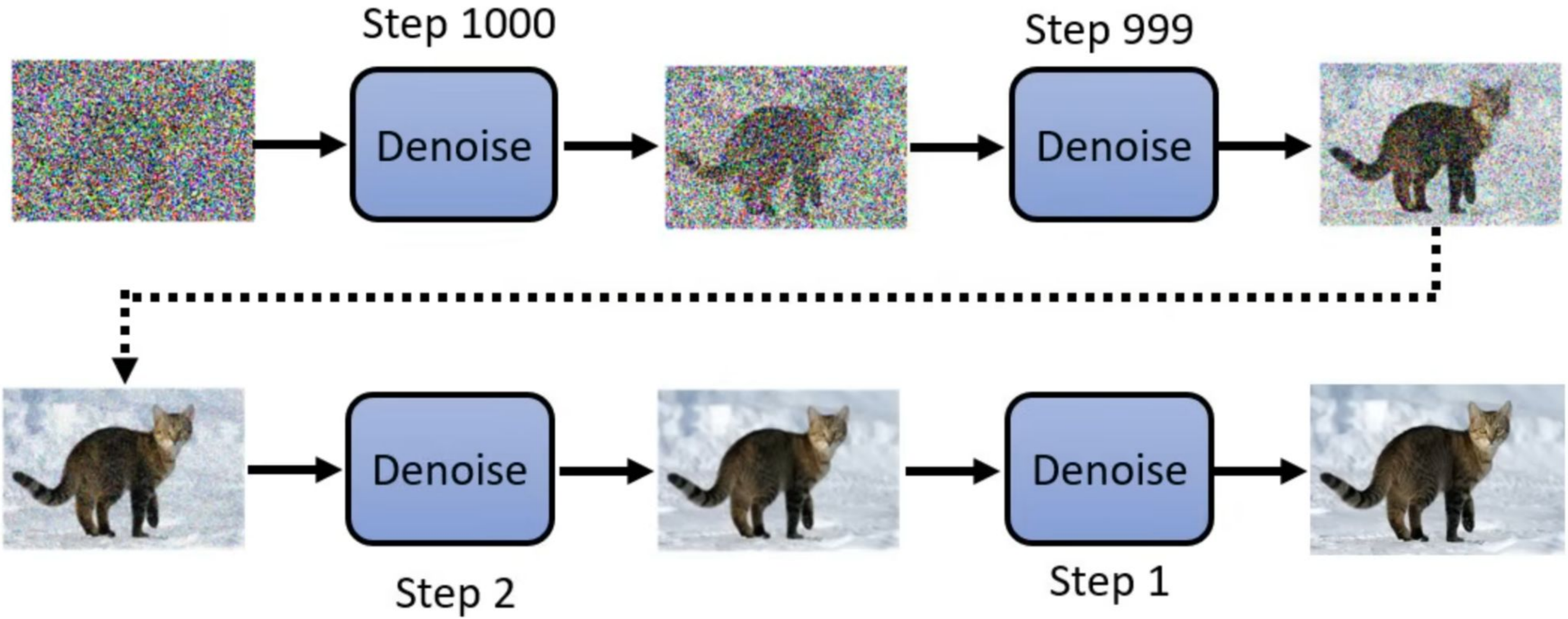
Fully connected output layer for class prediction

Evaluation

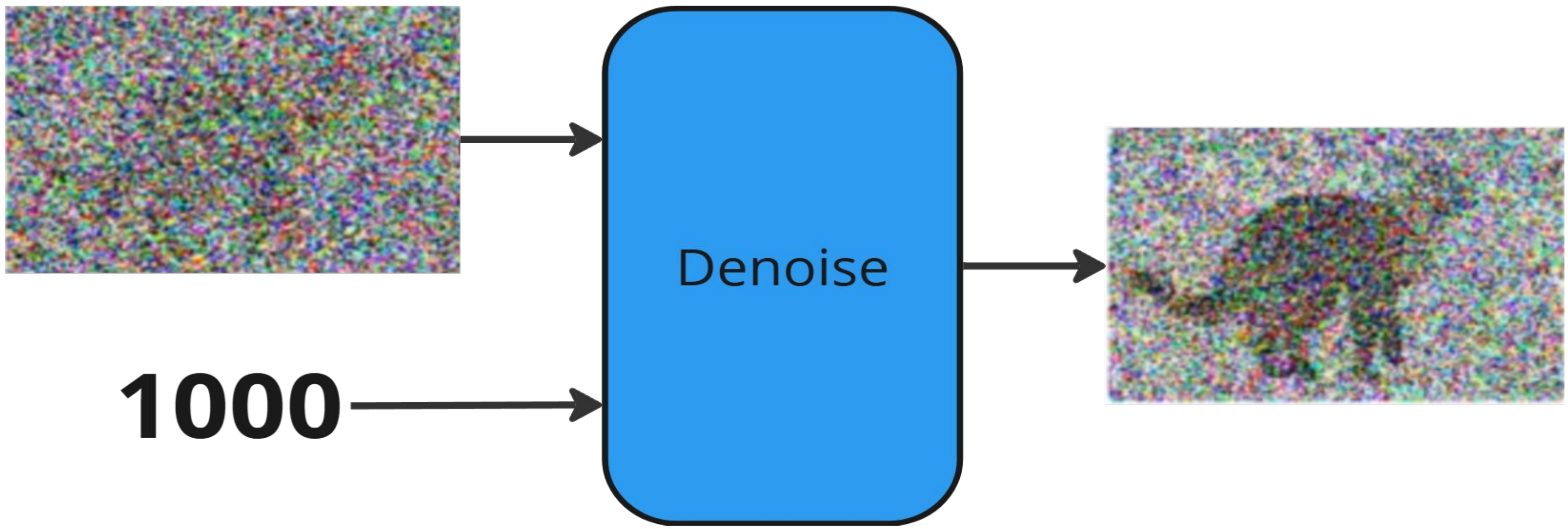
Accuracy, loss tracking, optional validation

Diffusion model

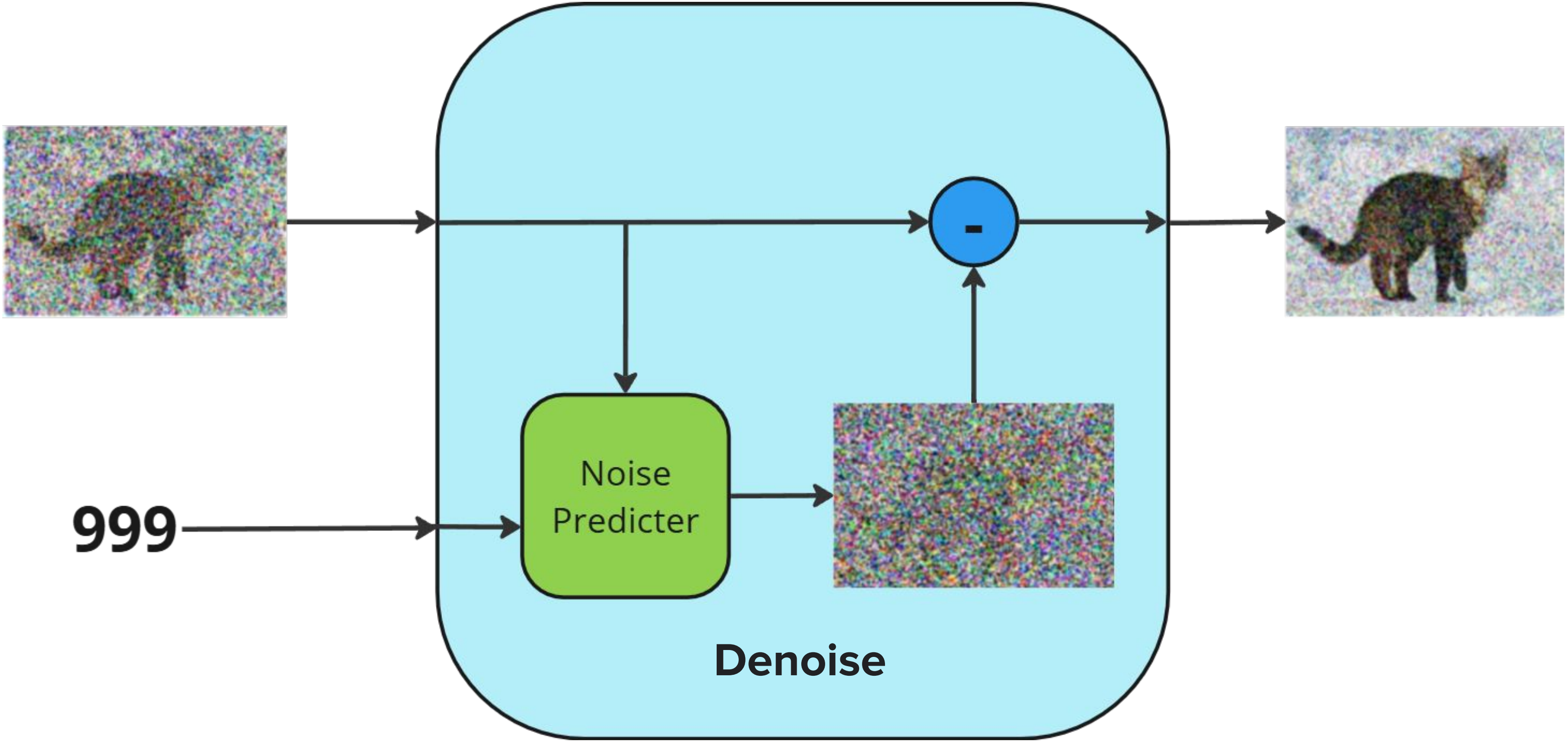
Inference


Step 1000
Denoise
Step 999
Denoise
Denoise
Step 2
Denoise
Step 1

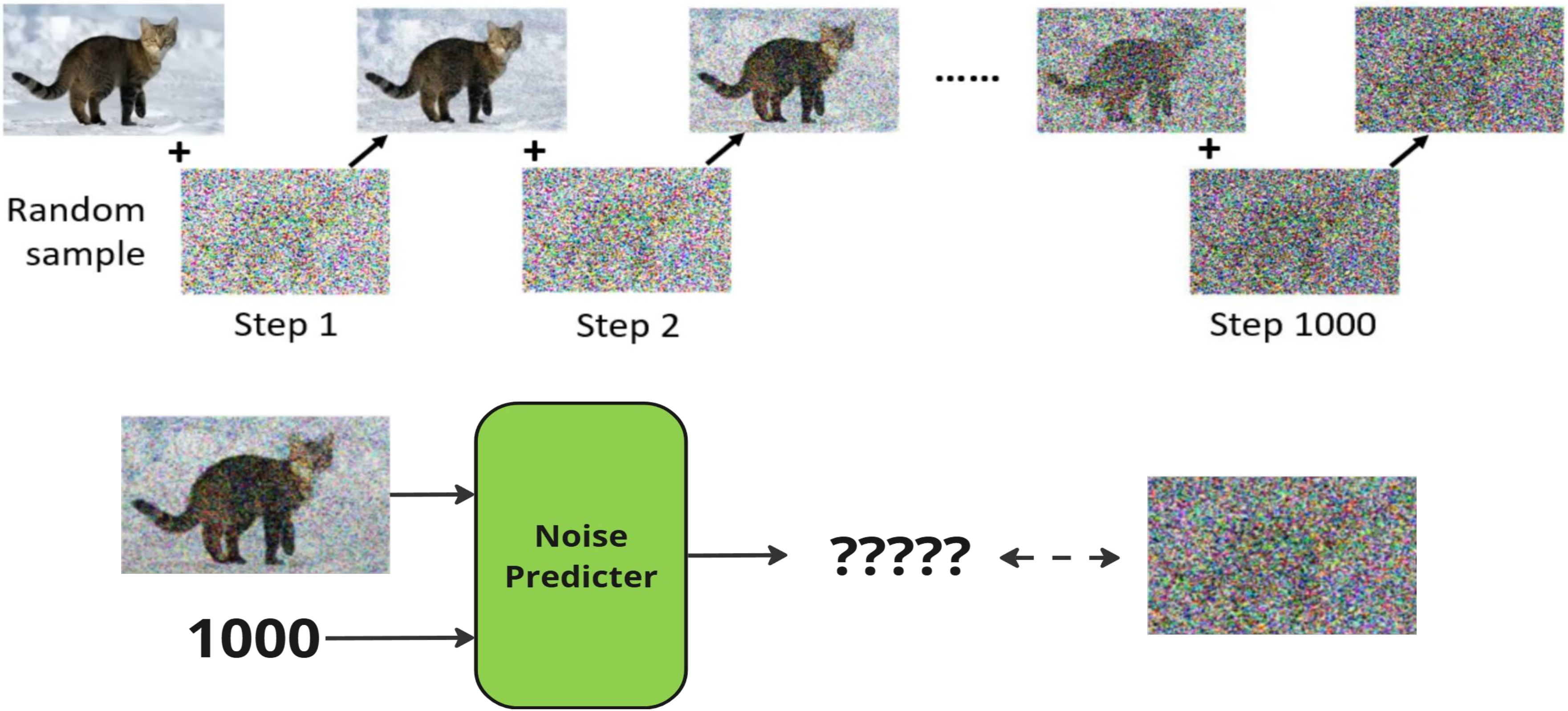
The Input of Diffusion Model



The Inside of Diffusion Model



Training Step



Denoising Diffusion Probabilistic Models

Algorithm 1 Training

repeat

$\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$

$\quad t \sim \mathrm{Uniform}(\{1, \ldots, T\})$

$\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\quad$ Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

until converged

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

for $t = T, \ldots, 1$ **do**

$\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$

$\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$

end for

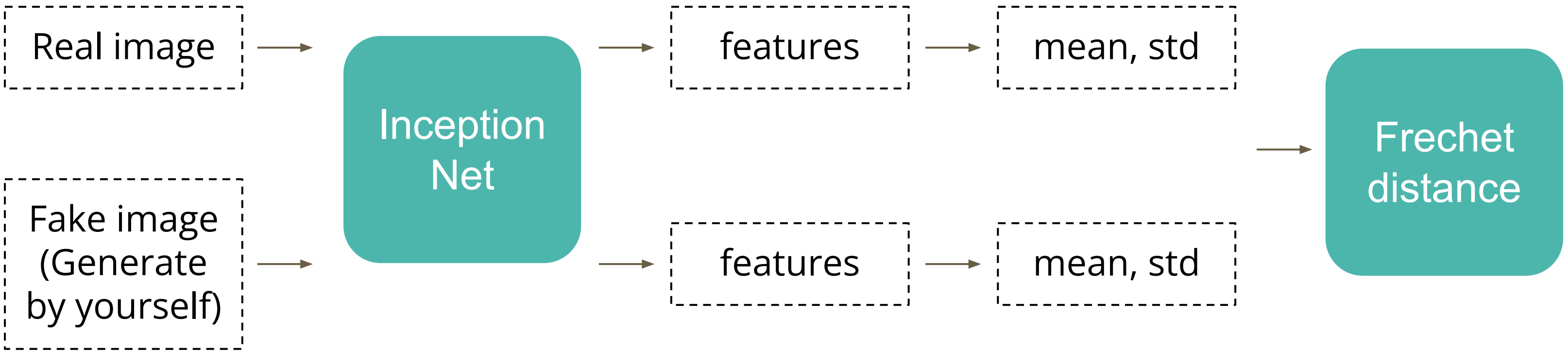
return $\mathbf{x}_0$

https://arxiv.org/pdf/2006.11239
https://youtu.be/azBugJzmz-o?si=D2K8NWF-5rSUYJ7R

Fréchet Inception Distance

- FID does not directly compare image pixels. Instead, it uses a pre-trained model (usually **Inception v3**) to extract high-level features from the images, and then compares the distributions of these features.

$$FID(x,g) = \left|\mu_x - \mu_g\right|^2 + Tr\left(\Sigma_x + \Sigma_g - 2\sqrt{\Sigma_x \Sigma_g}\right)$$



Diffusion code

https://colab.research.google.com/drive/1F6pkuAADeOWzdVa9zlmMmm-VYKDEf-oZ?usp=sharing

How to Fine-Tune Llama 3.2 Vision (11B)

Sample Code