MACHINE LEARNING IN PYTHON

(PART 4):

DIFFUSION MODELS IN PYTORCH

LUKE SHENEMAN
GENERATIVE ARTIFICIAL INTELLIGENCE

Text to Image (Stable Diffusion)

Text to Video

ChatGPT passes medical, law, and business exams
The AI test generator has educators debating its role in the classroom.

Real estate agents say they can’t imagine working without ChatGPT now
By Samantha Murphy Kelley, OW Business
updated 11:57 AM, Fri, January 28, 2023

Google has developed a music-making AI bot
The AI takeover hits the music industry.
By Chase Oliphant
January 30, 2023

Microsoft’s new AI can simulate anyone’s voice with 3 seconds of audio
Text-to-speech model can preserve speaker’s emotional tone and acoustic environment.

Denoising EEG Signals for Real-World BCI Applications Using GANs
Eamonn Boyle, Peter Redmond, Andrew Casey, Martina De Vries, Geraldine Boyle

Generative adversarial networks unlock new methods for cognitive science

19 Ways to Use ChatGPT in Your Classroom

ChatGPT Is Making Universities Rethink Plagiarism
Students and professors can’t decide whether the AI chatbot is a research tool—or a cheating engine.
Generative AI: Learn a *latent* representation of the distribution of our complex training data and then sample from it.

Deep Learning

Training Data

Diffusion, etc.

Transformers, etc.

Training data
(e.g. 64x64x3~12K dims)

Sampling

$p(x)$

$\hat{p}(x)$

Learning
DIFFUSION MODELS
CONDITIONING IMAGE GENERATION

• Provide natural language text prompts to guide reverse diffusion process

• Text-to-Image Diffusion Models are both:
  • Image Generation Models
  • Language Models
"A person half Yoda half Gandalf"
OVERVIEW

• Recap from Parts 1-3
  • Machine Learning Basics
  • Neural Networks
• Tensors
• Convolutional Neural Networks (CNNs)
• GPUs and CUDA
• PyTorch
  • Why use PyTorch?
  • Implementing a Diffusion Model in Python
  • Train and Test our Diffusion Model
REVIEW OF BASICS

- Machine learning is a data-driven method for creating models for prediction, optimization, classification, generation, and more

- Python and scikit-learn

- MNIST

- Artificial Neural Networks (ANNs)
**NEURAL NETWORK BASICS**

```python
from sklearn.neural_network import MLPClassifier

# Instantiate a Neural Network and train our model
nn = MLPClassifier(hidden_layer_sizes=(50,25),
                    max_iter=50,
                    n_iter_no_change=50,
                    activation = 'relu',
                    solver='adam',
                    random_state=42,
                    verbose=True)
nn.fit(X_train, y_train)
```

### Activation Functions

- **Sigmoid**
  \[
  \sigma(x) = \frac{1}{1+e^{-x}}
  \]

- **Leaky ReLU**
  \[
  \max(0.1x, x)
  \]

- **tanh**
  \[
  \tanh(x)
  \]

- **Maxout**
  \[
  \max(w_1^T x + b_1, w_2^T x + b_2)
  \]

- **ReLU**
  \[
  \max(0, x)
  \]

- **ELU**
  \[
  \begin{cases} 
  x & x \geq 0 \\
  \alpha(e^x - 1) & x < 0
  \end{cases}
  \]

Weights and Biases
A tensor is an N-dimensional array of data

- Rank 0 Tensor: scalar
- Rank 1 Tensor: vector
- Rank 2 Tensor: matrix
- Rank 3 Tensor
Images are tensors!
We need image *filters* to help us extract features
EXAMPLE: SOBEL FILTER

Sobel kernels =

\[
\begin{array}{ccc}
-1 & 0 & +1 \\
-2 & 0 & +2 \\
-1 & 0 & +1 \\
\end{array}
\]

\[
\begin{array}{ccc}
+1 & +2 & +1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{array}
\]

Gx  Gy
CONVOLUTIONAL NEURAL NETWORK

**INPUT**

(28 x 28 x 1)

**Conv_1**
Convolution
(5 x 5) kernel
valid padding

**Max-Pooling**
(2 x 2)

**Conv_2**
Convolution
(5 x 5) kernel
valid padding

**Max-Pooling**
(2 x 2)

**fc_3**
Fully-Connected
Neural Network
ReLU activation

**fc_4**
Fully-Connected
Neural Network
(with dropout)

**OUTPUT**

(4 x 4 x n2)

n3 units

n2 channels
(8 x 8 x n2)

n2 channels
(12 x 12 x n1)

n1 channels
(24 x 24 x n1)

n1 channels
(28 x 28 x 1)
Moore's Law for CPUs is Dead

"Moore's Law for CPUs is Dead"
WHY GPUs EXACTLY?

- CNNs are all about matrix and vector operations (multiplication, addition)
- GPUs can perform parallel multiplication and addition steps per each clock cycle.
DIFFUSION MODELS
FORWARD DIFFUSION

- Define how many **time steps** will be used (common to use hundreds or more)
- Establish a **noise schedule** which describes the rate at which Gaussian noise is added

I used 100 timesteps. Larger models like Stable Diffusion use thousands of smaller steps.
I used a cosine noise schedule.
TIME STEP ENCODING

I encode timestep as another band in the image in pixel-space

RGB Image + Integer timestep = 4-Channel RGB + Timestep
Possible loss functions for our U-Net

\[
\begin{align*}
\text{loss}_1 &= \text{MSE}(\text{pred}_t, \text{original}) \\
\text{loss}_2 &= \text{MSE}(\text{pred}_t, \text{noisy}_{t-1}) \\
\text{loss}_3 &= \text{pred}_t - \text{noisy}_t
\end{align*}
\]

Other Hyperparameters:

- Epochs = 100
- Timesteps = 100
- Batch Size = 1250
- Optimizer = Adam
- Learning Rate = 0.001
Core Training Loop

```python
schedule = cosine_schedule(TIMESTEPS)
for each Epoch:
    for each Batch b:
        for each Timestep t:
            img = add_gaussian_noise(img, schedule(t))
            predicted = UNet(img)
            loss = loss_function(img, predicted)
            backward_propagation and optimization
```
CELEB FACES ATTRIBUTES (CELEBA) DATASET

• 202,599 number of face images of various celebrities
• 10,177 unique identities, but names of identities are not given
• 40 binary attribute annotations per image
• 5 landmark locations

Images “in the wild” or Cropped/Aligned
SOME PRELIMINARY OUTPUT

Oh no!
Use separate AI model for *upsampling*

https://github.com/twtygqyy/pytorch-SRResNet
Might not be terrific, but…

It was trained on only 5000 images for a few hours on a single RTX 4090 GPU

Stable Diffusion was trained on 600 million captioned images

Took 256 NVIDIA A100 GPUs on Amazon Web Services a total of 150,000 GPU-hours
At a cost of $600,000
Conditioning reverse Diffusion on Text prompts
PRE-PROCESSING CELEBA DATASET

• Read first 5000 annotations into PANDAS dataframe (easy!)
• For each image, get the heading names for positive attributes
• Convert heading names into a text prompt:
  e.g. “Photo of person <attribute_x>, <attribute_y>, <attribute_z>, ...”
  e.g. “Photo of person bushy eyebrows, beard, mouth slightly open, wearing hat.”
• Crop the largest square from the image, then resize to 64x64x3 numpy array
• Use OpenAI CLIP model to find the image embeddings and text embeddings for every image/prompt pair.
• Create a 5000 element Python list of 4-tuples:
  • (filename, 64x64xRGB image_array, image_embedding, prompt_embedding)
• Pickle list to a file we can quickly load into memory when we train our model!
OPENAI CLIP MODEL (CONTRASTIVE LANGUAGE–IMAGE PRE-TRAINING)

- Open source/weights multi-modal AI model trained on image, caption pairs
- Shared embedding space!
- Use transformer model (GPT-2) to create token embeddings from text
- Use vision transformer (VIT) to create token embeddings from images
CLIP Examples

https://openai.com/research/clip

healthy lymph node tissue (77.2%) Ranked 2 out of 2 labels

✓ this is a photo of healthy lymph node tissue

❌ this is a photo of lymph node tumor tissue

a photo of guacamole, a type of food.

❌ a photo of ceviche, a type of food.

✓ a photo of edamame, a type of food.

✓ a photo of tuna tartare, a type of food.

✓ a photo of hummus, a type of food.

a photo of a 2012 honda accord coupe.

❌ a photo of a 2012 honda accord sedan.

❌ a photo of a 2012 acura tl sedan.

✓ a photo of a 2012 acura tsx sedan.

✓ a photo of a 2008 acura tl type-s.

✓ a photo of country line dancing.

❌ a photo of square dancing.

تنسيق: a photo of swing dancing.

✓ a photo of dancing charleston.

✓ a photo of salsa dancing.
import torch
import clip
from PIL import Image

device = "cuda" if torch.cuda.is_available() else "cpu"
model, preprocess = clip.load("ViT-B/32", device=device)

image = preprocess(Image.open("CLIP.png")).unsqueeze(0).to(device)
text = clip.tokenize(["a diagram", "a dog", "a cat"]).to(device)

with torch.no_grad():
    image_features = model.encode_image(image)
text_features = model.encode_text(text)

    logits_per_image, logits_per_text = model(image, text)
    probs = logits_per_image.softmax(dim=-1).cpu().numpy()

print("Label probs:", probs)  # prints: [[0.9927937 0.00421068 0.00299572]]

https://github.com/openai/CLIP

Zero-shot classifications!
Conditioning Generative AI (DALL-E)
Generating captions for images or video
Image similarity search
Content Moderation
Object Tracking
CLIP uses vectors with 512 dimensions

GPT3 (Davinci) uses 12888 dimensions

Vector embeddings capture the deeper semantic context of a word or text chunk...or image...or anything.

The semantics of an object are defined by its multi-dimensional and multi-scale co-occurrence and relationships with other objects in the training data.

Semantic vector embeddings are learned from vast amounts of data.

400,000,000 (image, text) pairs

CLIP was trained on 256 large GPUs for 2 weeks.
One way to train a multi-modal embedding layer

“A cute Welsh Corgi dog.”

Learning your semantic embeddings first (GPT)

Train your CNN to predict matching embedding vectors
How CLIP handle multi-model semantic embeddings

1. Contrastive pre-training

2. Create dataset classifier from label text

3. Use for zero-shot prediction
Broadcast Prompt CLIP embeddings into UNet. Vector = 512 length

Simply add embeddings to values along the feature map axis
Now for Code

github.com/sheneman/diffusion