R.M.K GROUP OF ENGINEERING INSTITUTIONS

R.M.K
GROUP OF
INSTITUTIONS

R.M.K
GROUP OF
INSTITUTIONS

R.M.K
GROUP OF
INSTITUTIONS

R.M.K
GROUP OF
INSTITUTIONS

Digital Course Material
22AI005
Introduction to Generative AI

Department : CSE(CYBER SECURITY)

Batch/Year : 2022-2026/IV

Created by : Ms G.K.MONICA

AP/CSE(CS)

Date : 07-07-2025

TABLE OF CONTENTS

R.M.K GROUP OF INSTITUTIONS

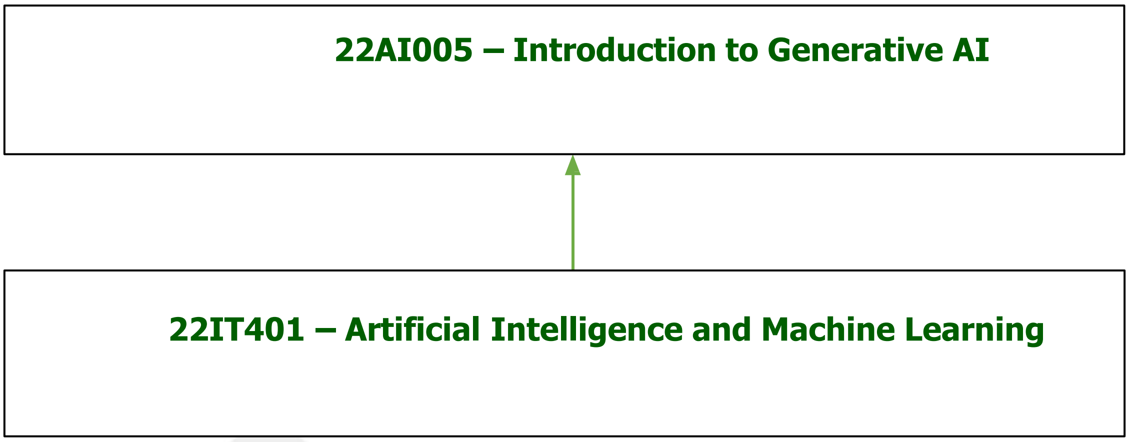
COURSE OBJECTIVES

The Course will enable learners to:

- To understand the basic concepts of Generative AI.
- To build Generative AI systems to generate images.
- To understand the concept used in Generative AI Models.
- To use various Generative AI models.
- To compare and use the various Large Language Models.
- To understand the basics of Prompt Engineering.

PRE REQUISITES

PRE-REQUISITE CHART


22AI005 – Introduction to Generative AI
22IT401 – Artificial Intelligence and Machine Learning

INTRODUCTION TO GENERATIVE AI L T P C

3 0 0 3

OBJECTIVES:

- To understand the basic concepts of Generative AI.
- To build Generative AI systems to generate images.
- To understand the concept used in Generative AI Models.
- To use various Generative AI models.
- To compare and use the various Large Language Models.
- To understand the basics of Prompt Engineering.

UNIT I INTRODUCTION 9

Generative Models – Image transformation – Challenges - Deep Neural Networks – Perceptron – back propagation – CNN – RNN – Optimizer.

UNIT II IMAGE GENERATION 9

Creating encodings of images – variational objective – Inverse Autoregressive flow – Importing CIFAR – Creating the network from TensorFlow 2.

UNIT III GENERATIVE ADVERSARIAL NETWORKS 9

Generative Adversarial Networks – Vanilla GAN – Improved GANs – Progressive GAN – Challenges – Paired style transfer – Unpaired style transfer – Deepfakes – Modes of operation – key feature set – High level flow – Replacement – Re-enactment.

UNIT IV LARGE LANGUAGE MODELS 9

Overview of LLMs - Transformers – GPT – Types of LLMs – Key concepts – other Transformers – T5 – Generative Pre-Training Models – Multi-modal Models – DALL.E 2

UNIT V PROMPT ENGINEERING 9

Basics – In-Context Learning – In-Context Prompting – Techniques – Image Prompting – Prompt Hijacking – Challenges.

TOTAL: 45 PERIODS

COURSE OUTCOME

Course Code	Course Outcome Statement	Cognitive/ Affective Level of the Course Outcome	Expected Level of Attainment
Course Outcome Statements in Cognitive Domain			
C305.1	Elaborate the basic concepts of Generative AI	Understand K2	60%
C305.2	Build Generative AI systems to generate images	Analyse K4	60%
C305.3	Apply the concepts used in Generative AI Models	Apply K3	60%
C305.4	Use various Generative AI models.	Apply K3	60%
C305.5	Compare and use the various Large Language Models	Analyse K4	60%
C305.6	Analyze the basics of Prompt Engineering.	Apply K3	60%
Course Outcome Statements in Affective domain			
C305.7	Attend the classes regularly	Respond (A2)	95%
C305.8	Submit the Assignments regularly.	Respond (A2)	95%
C305.9	Participation in Seminar/Quiz/ Group Discussion/ Collaborative learning and content beyond syllabus	Valuing (A3)	95%

CO-PO/PSO MAPPING

Correlation Matrix of the Course Outcomes to Programme Outcomes and Programme Specific Outcomes Including Course Enrichment Activities

Course Outcomes (Cos)		Programme Outcomes (POs), Programme Specific Outcomes (PSOs)														
		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
		K3	K4	K5	K5	K3/K5	A2	A3	A3	A3	A3	A3	A2	K3	K3	K3
C305.1	K3	3	2	1	1	3						3		3	3	3
C305.2	K4	3	3	2	2	3						3		3	3	3
C305.3	K2	2	1									2		3	3	3
C305.4	K3	3	2	1	1	3						3		3	3	3
C305.5	K4	3	3	2	2	3						3		3	3	3
C305.6	K3	3	2	1	1	3						3		2	2	2
C305.7	A2												3			
C305.8	A2								2	2	2		3			
C305.9	A3						3	3		3	3		3			
C305		3	3	2	2	3	1	1	1	3	3	3	3	3	3	3

R.M.K GROUP OF INSTITUTIONS

UNIT II

IMAGE GENERATION

LECTURE PLAN – UNIT II

	UNIT II IMAGE GENERATION						
Sl. No	TOPIC	NO OF PERIODS	PROPOSED LECTURE	ACTUAL LECTURE	PERTAINING CO(s)	TAXONOMY LEVEL	MODE OF DELIVERY
			PERIOD	PERIOD			
1	Creating encodings of images	1			CO2	K2	PPT
2	variational objective	1			CO2	K2	PPT
3	Inverse Autoregressive flow	1			CO2	K2	PPT
4	Importing CIFAR	1			CO2	K3	PPT
5	Creating the network from TensorFlow 2	1			CO2	K3	PPT

LECTURE PLAN – UNIT II

ASSESSMENT COMPONENTS	MODE OF DELEIVERY
AC 1. Unit Test	MD 1. Oral presentation
AC 2. Assignment	MD 2. Tutorial
AC 3. Course Seminar	MD 3. Seminar
AC 4. Course Quiz	MD 4 Hands On
AC 5. Case Study	MD 5. Videos
AC 6. Record Work	MD 6. Field Visit
AC 7. Lab / Mini Project	
AC 8. Lab Model Exam	
AC 9. Project Review	

ACTIVITY BASED LEARNING – UNIT II QUIZ- LINK

Unit II:

Inverse Autoregressive flow

https://testbook.com/objective-questions/mcq-on-type-of-flow--5eea6a0c39140f30f369e18e

Encodings of images

https://quizgecko.com/learn/image-compression-and-encoding-kk19ep

CIFAR

https://www.cs.toronto.edu/~kriz/cifar.html

TensorFlow 2:

https://www.acmecollinsschool.com/tensorflow-2-mcqs-questions.html

TEST YOURSELF

1.Which of the following techniques is used to ensure that the latent space in a Variational Autoencoder is well-behaved and follows a normal distribution?

A. Reconstruction loss

B. Cross-entropy loss

C. KL Divergence loss

D. Dropout regularization

2..What is the primary purpose of creating encodings of images in deep learning?

A. To improve image resolution

B. To reduce the dimensionality of the data while preserving important information

C. To change the color scheme of the image

D. To segment the image into different regions

3.What is the primary purpose of using Inverse Autoregressive Flow (IAF) in variational inference?

A. To reduce the computational complexity of training deep neural networks

B. To enhance the expressiveness of the approximate posterior distribution

C. To perform dimensionality reduction on input data

D. To visualize high-dimensional data

4.TensorFlow Data API does not support text files?

A. True

B. False

5.Using TensorFlow Data API we can do the following:

A. Batching

B. Prefetching

C. Gradient Descent

D. All of the above

UNIT II
Creating encodings of images

Creating encodings of images involves transforming images into a compact, fixed-size representation (or vector) that captures the essential features and information. These encodings are used in various applications such as image retrieval, classification, clustering, and as inputs to other machine learning models.

Overview

1. **Feature Extraction:** Extract meaningful features from the image using pre-trained convolutional neural networks (CNNs).
2. **Dimensionality Reduction (Optional):** Reduce the dimensionality of the extracted features to create a more compact encoding.
3. **Storage/Usage:** Use these encodings for downstream tasks or store them for later use.

Steps to Create Image Encodings

1. Feature Extraction

The most common approach is to use a pre-trained CNN, such as VGG16, ResNet, or Inception, to extract features from images. These models, trained on large datasets like ImageNet, can effectively capture general features.

Using Pre-trained Models:

```
import torch

import torch.nn as nn

from torchvision import models, transforms

from PIL import Image

# Load the pre-trained ResNet model

model = models.resnet50(pretrained=True)

# Remove the last fully connected layer

model = nn.Sequential(*list(model.children())[:-1])

# Set the model to evaluation mode

model.eval()
```

```
# Image transformations
preprocess = transforms.Compose
([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
def get_image_encoding(image_path):
 # Load the image
    image = Image.open(image_path)
    image = preprocess(image)
    image = image.unsqueeze(0)  # Add batch dimension
# Extract features
    with torch.no_grad():
    encoding = model(image)
# Flatten the encoding
    encoding = encoding.view(encoding.size(0), -1)
    return encoding.numpy()
# Example usage
image_path = 'path_to_image.jpg'
encoding = get_image_encoding(image_path)
print(encoding)
```

Dimensionality Reduction (Optional)

Depending on the application, the dimensionality of the extracted features might be reduced. Techniques like PCA (Principal Component Analysis) or t-SNE (t-Distributed Stochastic Neighbor Embedding) are commonly used.

Using PCA for Dimensionality Reduction:

From sklearn.decomposition import PCA

# Suppose encodings is a list of image encodings obtained from the above step

encodings = [get_image_encoding(img_path) for img_path in list_of_image_paths]

encodings = np.vstack(encodings) # Stack to a single array

# Apply PCA to reduce dimensionality

pca = PCA(n_components=128) # Example: Reduce to 128 dimensions

reduced_encodings = pca.fit_transform(encodings)

Storage/Usage

The encodings can be stored in a file or database for later retrieval or used directly in downstream machine learning tasks.

Storing Encodings:

import numpy as np

# Save to a file

np.save('image_encodings.npy', reduced_encodings)

# Load from a file

loaded_encodings = np.load('image_encodings.npy’)

Applications

1. **Image Retrieval:** Encodings can be used to find similar images by comparing their encodings using distance metrics like Euclidean distance.
2. **Classification:** Encodings can serve as inputs to classifiers (e.g., SVM, random forest) for tasks such as image classification.
3. **Clustering:** Encodings can be clustered using algorithms like K-means to group similar images.
4. **Anomaly Detection:** Encodings can help in identifying outliers or anomalies in image datasets.

variational objective

The objective of a Variational Autoencoder (VAE) is to learn a probabilistic latent space representation of the input data such that it can generate new data points similar to the input data. This is achieved through two main components: the encoder (which maps the input data to a latent space) and the decoder (which maps the latent space back to the data space).

The training objective of a VAE combines two loss terms:

1. **Reconstruction Loss**: This term measures how well the decoder reconstructs the input data from the latent representation. It is typically measured using Mean Squared Error (MSE) or Binary Cross-Entropy (BCE) between the original input data and the reconstructed data. The reconstruction loss ensures that the latent space captures the important features of the input data.
2. **KL Divergence Loss**: This term measures how close the learned latent distribution q(z|x) (produced by the encoder) is to the prior distribution p(z). The prior is usually a standard normal distribution N(0,I. The KL Divergence encourages the latent representations to be similar to this prior distribution, which helps in regularizing the latent space and ensuring that the generated data points are diverse and meaningful.

Mathe

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\phi(z|x) \| p(z))$$

Where

- Θ are the parameters of the decoder.
- Φ are the parameters of the encoder.
- x is the input data.
- z is the latent representation.
- pθ(x|z) is the likelihood of the data given the latent variable.
- qϕ(z|x) is the approximate posterior distribution (encoder).
- p(z) is the prior distribution over the latent variables.

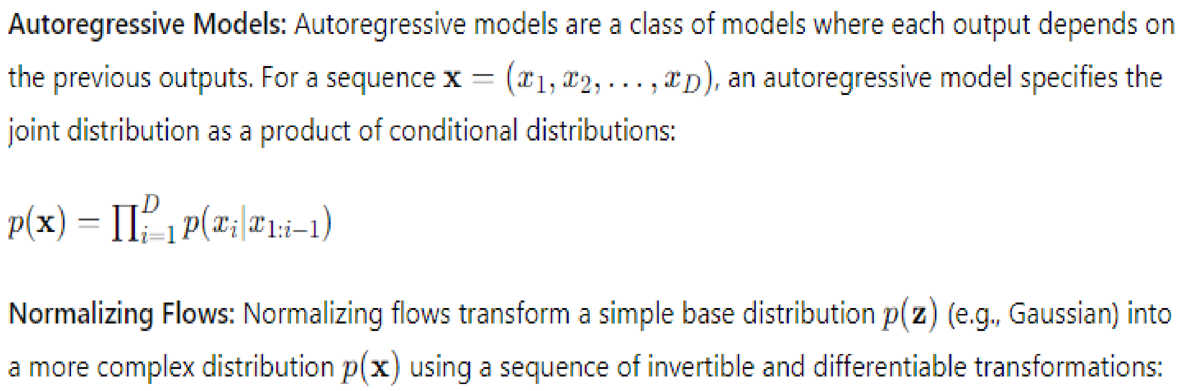
Explanation

- **Reconstruction** $(\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)])$: This term ensures that the decoded output (reconstructed input) is as close as possible to the original input. It forces the decoder to learn to generate realistic data from the latent representations $(\mathrm{KL}(q_\phi(z|x) \| p(z)))$:
- **KL Divergence Term** This term regularizes the latent space by ensuring that the distribution of the latent variables zzz stays close to the prior distribution p(z)p(z)p(z), usually a standard Gaussian. This regularization helps in generating new, diverse data samples and prevents overfitting.

By balancing these two terms, the VAE ensures that the latent space is both informative for reconstructing the data and well-regularized to allow for meaningful sampling and generation of new data points.

Inverse Autoregressive flow

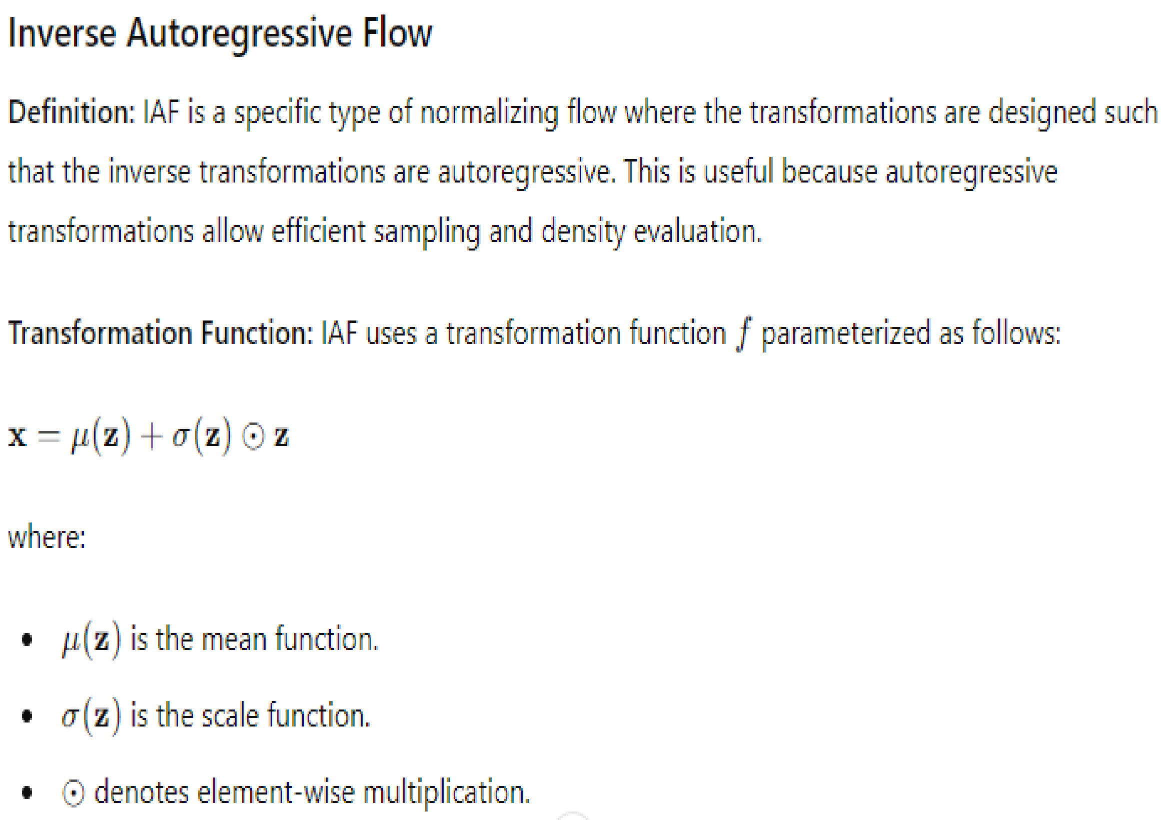
Inverse Autoregressive Flow (IAF) is a powerful technique used in the field of probabilistic machine learning, particularly for variational inference and generative modeling. It allows for the construction of complex, flexible distributions by transforming simple base distributions through a sequence of invertible transformations.



Autoregressive Models: Autoregressive models are a class of models where each output depends on the previous outputs. For a sequence $\mathbf{x} = (x_1, x_2, \ldots, x_D)$, an autoregressive model specifies the joint distribution as a product of conditional distributions:

$$p(\mathbf{x}) = \prod_{i=1}^{D} p(x_i | x_{1:i-1})$$

Normalizing Flows: Normalizing flows transform a simple base distribution $p(\mathbf{z})$ (e.g., Gaussian) into a more complex distribution $p(\mathbf{x})$ using a sequence of invertible and differentiable transformations:



Inverse Autoregressive Flow

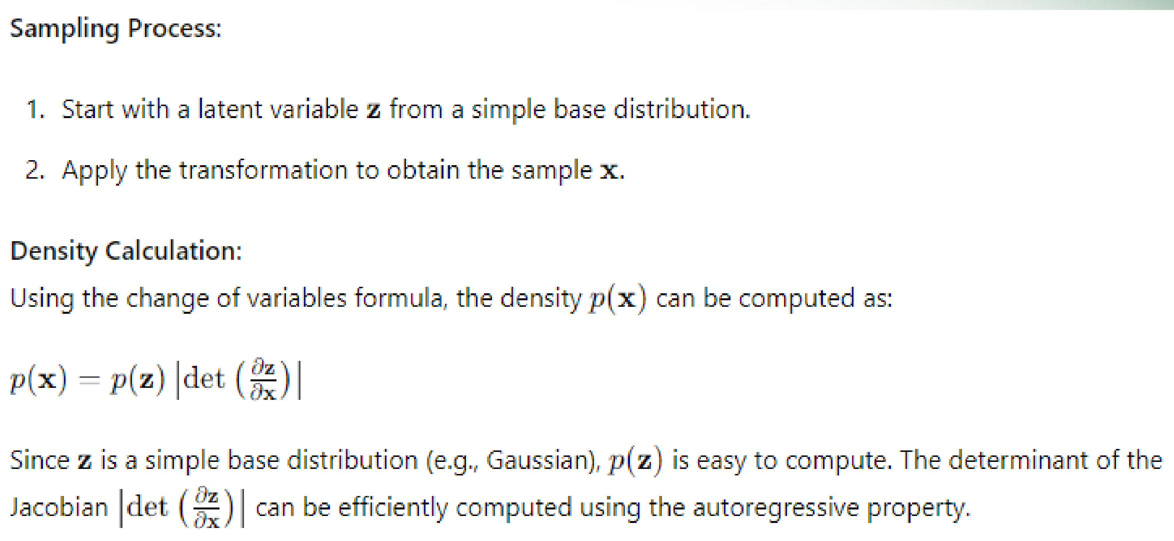
Definition: IAF is a specific type of normalizing flow where the transformations are designed such that the inverse transformations are autoregressive. This is useful because autoregressive transformations allow efficient sampling and density evaluation.

Transformation Function: IAF uses a transformation function f parameterized as follows:

$$\mathbf{x} = \mu(\mathbf{z}) + \sigma(\mathbf{z}) \odot \mathbf{z}$$

where:

- $\mu(\mathbf{z})$ is the mean function.
- $\sigma(\mathbf{z})$ is the scale function.
- $\odot$ denotes element-wise multiplication.


Sampling Process:

1. Start with a latent variable $\mathbf{z}$ from a simple base distribution.
2. Apply the transformation to obtain the sample $\mathbf{x}$.

Density Calculation:
Using the change of variables formula, the density $p(\mathbf{x})$ can be computed as:

$$p(\mathbf{x}) = p(\mathbf{z}) \left|\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)\right|$$

Since $\mathbf{z}$ is a simple base distribution (e.g., Gaussian), $p(\mathbf{z})$ is easy to compute. The determinant of the Jacobian $\left|\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)\right|$ can be efficiently computed using the autoregressive property.

Key Properties

1.Invertibility: The transformation is designed to be invertible, ensuring that both sampling and density evaluation are feasible.

2.Expressiveness: IAF can represent complex distributions due to the flexibility of the transformation functions μ\muμ and σ\sigmaσ.

3.Autoregressive Structure: The autoregressive nature allows efficient computation of the Jacobian determinant, crucial for density estimation.

Applications

1.Variational Inference: IAF is often used in Variational Autoencoders (VAEs) to approximate the posterior distribution. The flexibility of IAF allows for better approximation of complex posterior distributions.

2.Generative Modeling: In models like PixelCNN, IAF helps in generating high-quality samples by modeling complex distributions of pixel values.

Implementation Considerations

•**Neural Networks for μ\muμ and σ\sigmaσ:** Typically, neural networks parameterize the functions μ\muμ and σ\sigmaσ.

•**Efficiency:** While IAF can be computationally intensive due to the need for sequential transformations, it is still efficient compared to other complex distribution modeling techniques.

Creating the network from TensorFlow 2

A neural network comprises several nodes in each layer of the neural network architecture which are interconnected. They are important for learning new features from instances of the data. Thus, they prove to be more accurate than most machine learning algorithms like logistic regression, support vector machine, etc. They can be used for processing images and identifying the objects within them. There are many other use cases of neural networks where they have proved to be very accurate. In this article, we'll be focusing on TensorFlow neural networks and how to build them.

What is TensorFlow?

TensorFlow is an open-source library developed in Python for the implementation of machine learning and artificial intelligence. It was initially developed for the Python language but the support has now extended to other languages like Java, C++, and JavaScript. The library was built by Google Brain with focus on implementing deep neural networks.

Some of the important features of TensorFlow are:

1. Automatic differentiation: A process that calculates the gradient vector of each of the parameters in the model that help in optimizing the performance of the model, such as backpropagation.

2. Eager execution: A mode that directly executes the code rather than adding to a computational graph that executes the code at a later stage.

3. Distribute: An API that distributes the computation of models on various devices which results in speeding up the computation process.

4. Losses: Functions that are required for assessing a trained model. They compare the model's output with the expected output and compute the error loss. A few popular loss functions are binary cross entropy and mean squared error.

Setting up TensorFlow

To install the latest version of TensorFlow, the following system requirements must be met:

1. 64-bit operating system
2. Python version 3.7 or later
3. Windows 7 or later
4. macOS 10.12.6 (Sierra) or later (no GPU support)
5. Ubuntu 16.04 or later

The TensorFlow library can be installed using the following commands:

```
# Ugrade pip to latest version

pip install --upgrade pip

#Installation using pip command

pip install tensorflow
```

There is another way of using TensorFlow too: with Docker. The only requirement is that the docker should be installed on the PC. You can run the following command in the terminal:

```
docker pull tensorflow/tensorflow:latest  # Download latest stable image

docker run -it -p 8888:8888 tensorflow/tensorflow:latest-jupyter  # Start Jupyter server
```

Developing a TensorFlow neural network

This section of the article will discuss some basic steps for developing a neural network using TensorFlow. Before this, we will look at basic terminologies related to convolutional neural networks (CNNs):

1. CNNs are a type of artificial neural network (ANN) which are very popular among machine learning practitioners. They consist of several layers, such as the convolutional layer, pooling layer, and output layer.
2. The convolutional layer is added to the architecture to extract useful features of image input.
3. The feature extraction in the previous step is achieved with the help of kernels. Kernels are N x N matrices that slide over the entire input image to extract the important features of an input image.
4. Pooling is required to reduce the dimensionality of the feature matrix. Sometimes, the feature matrix obtained after the convolutional layer results in a large number of features that need to be reduced so that it becomes computationally less expensive.
5. Padding is another type of layer present within CNN architecture. Here, additional zeroes are included to the edges of the matrix in order to read the edges precisely or get the output image similar to the one that was passed as the input image.
6. Once the matrix has been flattened by the pooling layer, the flattened vector is passed on to the fully connected layer. This layer is a neural network where weights are adjusted.

We now move on to the implementation of a TensorFlow neural network.

Importing CIFAR

The CIFAR-10 dataset is readily accessible in Python through the Keras library, which is part of TensorFlow, making it a convenient choice for developers and researchers working on machine learning projects, especially in image classification. In this article, we will explore CIFAR10 (classification of 10 image labels) from Keras/tensorflow.

What is the CIFAR10 Datasets?

The CIFAR-10 dataset contains 60,000 32×32 color images in 10 different classes, such as airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

CIFAR10 dataset consists of black and white images categorized into 10 types of clothing items, each represented by an integer label ranging from 0 to 9. This structure ensures clarity and organization in the data, facilitating effective classification tasks. CIFAR-10 dataset is a collection of images that are commonly used to train machine learning and computer vision algorithms. It is one of the most widely used datasets for machine learning research.

Full Form of CIFAR10 DataSet

The CIFAR-10 dataset stands for Canadian Institute For Advanced Research Dataset, where 10 stands for the count of representation classes, as discussed above.

Characteristics of CIFAR10 Dataset

The common characteristics of CIFAR10 dataset include:

- Number of Instances: 60,000 images
- Training Set:
  - ❑ 50,000 images
  - ❑ Each image is a 32×32 color image (RGB), resulting in a shape of (32, 32, 3).
  - ❑ Images are divided into 10 classes, with 5,000 images per class.
- Test Set:
  - ❑10,000 images
  - ❑Same structure as the training set, with 1,000 images per class.
- Pixel Values: Each pixel value (0-255) represents the grayscale intensity of the corresponding pixel in the image.
- Target: Target Column represents the type of clothing item (0-9)
- Number of Attributes: 1 (32×32 pixels = 1024 pixels

Structure of the CIFAR10 dataset:

(x_train, x_test): These variables contain the pixel data for the images.

x_train is the training set of the images, and

x_test is the testing set.

The images are 32×32 pixels in size and are represented as a numpy array of shape (32, 32, 3), where 3 stands for the three color channels (RGB).

(y_train, y_test): These are the corresponding labels for the images. Each label is an integer from 0 to 9, representing the class of representation, i.e.:

(Label) -> (Class)

0 -> Airplane

1 -> Automobile

2 -> Bird

3 -> Cat

4 -> Deer

5 -> Dog

6 -> Frog

7 -> Horse

8 -> Ship

9 -> Truck

How to Load CIFAR10 Datasets in Keras?

To load the CIFAR-10 dataset using Keras, you can use the CIFAR10 module from tensorflow.keras.datasets.

Syntax:

from tensorflow.keras.datasets import cifar10

# Load the CIFAR-10 dataset

(x_train, y_train), (x_test, y_test) = cifar10.load_data()

Significance of CIFAR10 in Machine Learning

The CIFAR-10 dataset holds significant importance in the field of machine learning for several reasons:

Benchmark Dataset: CIFAR-10 serves as a benchmark dataset for testing the performance of various machine learning algorithms, particularly in the domain of computer vision. Its popularity stems from its moderate size, making it suitable for experimentation and benchmarking without requiring extensive computational resources.

Real-World Image Classification: The CIFAR-10 dataset consists of 60,000 32×32 color images across 10 classes, with each class representing a different object category (e.g., airplane, automobile, bird, cat, etc.). This diversity makes CIFAR-10 a suitable dataset for training and evaluating image classification models on real-world, diverse image data.

Transfer Learning and Pre-Trained Models: CIFAR-10 is often used for transfer learning experiments, where models pre-trained on larger datasets (e.g., ImageNet) are fine-tuned on CIFAR-10 to adapt them to specific classification tasks. This approach leverages the learned representations from large-scale datasets to improve performance on smaller datasets like CIFAR-10.

Complexity: Despite its small size and relatively low resolution, CIFAR-10 remains a challenging dataset for machine learning models due to the variety of object classes, background clutter, and variations in object appearance and orientation within each class.

Applications of the CIFAR10 Dataset:

The CIFAR-10 dataset, with its collection of 60,000 images across 10 different classes (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks), serves as a fundamental resource for various applications and research in the field of computer vision and machine learning. Here are some key applications and uses of the CIFAR-10 dataset:

Benchmarking Models: CIFAR-10 is widely used to benchmark the performance of image recognition algorithms and neural network architectures. It helps researchers and developers compare the efficacy of different models under consistent conditions.

Training Convolutional Neural Networks (CNNs): Due to its moderate complexity and size, CIFAR-10 is excellent for training CNNs from scratch. It allows for rapid experimentation with network architectures, hyperparameters, and training procedures without the computational expense required for larger datasets like ImageNet.

Pre-training for Transfer Learning: CIFAR-10 can be used for pre-training models that are then fine-tuned on more specialized or smaller datasets. This is particularly useful when computational resources are limited or when the target dataset is too small to train a deep network effectively from scratch.

Educational Purposes: CIFAR-10 is commonly used in academic courses and tutorials related to machine learning and computer vision. It is complex enough to teach nuanced concepts of deep learning, yet simple enough for educational use.

Feature Learning: Researchers use CIFAR-10 to develop and test algorithms for learning feature representations from images. These learned features can be crucial for tasks such as image retrieval, classification, and anomaly detection.

Development of New Algorithms: Beyond traditional image classification, CIFAR-10 is used to develop new types of learning algorithms, such as semi-supervised learning, unsupervised learning, and self-supervised learning methods.

Real-time Object Recognition: Models trained on CIFAR-10 can be adapted to work in real-time applications, such as video surveillance and autonomous vehicles, where recognizing objects quickly and accurately is critical.

The CIFAR-10 dataset, readily accessible through the Keras library in Python, is a cornerstone in the realm of machine learning and computer vision. With its collection of 60,000 32×32 color images across 10 distinct classes, CIFAR-10 serves as a fundamental resource for various applications and research endeavors.

ASSIGNMENT – UNIT II

1. Explain the purpose of creating image encodings in the context of deep learning.(K6)
2. Describe the two main components of the loss function in a Variational Autoencoder (VAE)(K5)
3. What is the role of Inverse Autoregressive Flow (IAF) in enhancing the expressiveness of the variational posterior in VAEs.(K4)
4. How do you import and preprocess the CIFAR-10 dataset using TensorFlow 2(K6)
5. Outline the steps to build a simple Convolutional Neural Network (CNN) in TensorFlow 2 for encoding images.(K3)

PART A- UNIT-II

1.What is an image encoding in the context of deep learning?

Answer: An image encoding is a compact, lower-dimensional representation of an image that retains important features necessary for various tasks like classification, generation, or clustering.

2.**Compare the effectiveness of image encodings with raw pixel data for image classification tasks.**

Answer: Image encodings are generally more effective than raw pixel data because they abstract and condense the most important features, reducing noise and irrelevant information, which helps in improving the model's performance and training efficiency.

3.What are the two main components of the variational objective in VAEs?

Answer: The two main components are the reconstruction loss and the KL divergence loss.

4.Design an experiment to test the effect of different KL divergence weights on the performance of a VAE.

Answer: Train multiple VAEs on the same dataset with varying KL divergence weights. Evaluate and compare their reconstruction errors and the quality of generated samples using metrics like Mean Squared Error for reconstructions and visual inspection or FID scores for generated samples.

5.What is the purpose of Inverse Autoregressive Flow in variational inference?

Answer: The purpose of IAF is to enhance the expressiveness of the variational posterior by applying a sequence of invertible transformations to the latent variables, making the posterior distribution more flexible and capable of capturing complex dependencies.

6. Evaluate the benefits and potential drawbacks of using IAF in a VAE.

Answer: Benefits of using IAF include a more flexible and expressive posterior distribution, leading to better model performance in terms of reconstruction and generation. Potential drawbacks include increased computational complexity and training time, as well as the need for careful tuning of the model's hyperparameters.

7.What is the CIFAR-10 dataset?

Answer: The CIFAR-10 dataset is a collection of 60,000 32x32 color images in 10 different classes, with 6,000 images per class, commonly used for training machine learning and computer vision models.

8.What are the main components of a Convolutional Neural Network (CNN)?

Answer: The main components of a CNN are convolutional layers, pooling layers, and fully connected (dense) layers. Convolutional layers detect features, pooling layers reduce dimensionality, and dense layers perform classification or regression tasks..

9.Explain how pooling layers contribute to the functionality of a CNN.

Answer: Pooling layers contribute to the functionality of a CNN by reducing the spatial dimensions of the feature maps, which helps in lowering the computational load, reducing overfitting, and providing a form of spatial invariance to the model.

10.Analyze the impact of using different activation functions in the convolutional layers of a CNN.

Answer: Different activation functions impact the CNN's performance by affecting the learning capacity and non-linearity of the model. ReLU is commonly used due to its simplicity and effectiveness, but other functions like Leaky ReLU or ELU might provide better performance by addressing issues like dying neurons or vanishing gradients.

PART B- UNIT II

1. Explain why it is necessary to normalize pixel values when importing the CIFAR-10 dataset for deep learning tasks.(K3)
2. Compare the computational advantages and disadvantages of using different activation functions (e.g., ReLU, sigmoid, tanh) in the layers of a CNN designed for CIFAR-10 classification.(K4)
3. Describe how the loss function is constructed to balance reconstruction accuracy and latent space regularization.(K2)
4. Design a Variational Autoencoder (VAE) architecture with TensorFlow 2 that incorporates a variational objective.(K3)
5. Critically evaluate the role of Inverse Autoregressive Flow (IAF) in enhancing the flexibility of the approximate posterior distribution in a VAE. (K5)

SUPPORTIVE ONLINE COURSES – UNIT II

- https://www.classcentral.com/course/image-compression-generation-vae-19624
- https://arxiv.org/abs/1606.04934
- https://medium.com/fenwicks/tutorial-2-94-accuracy-on-cifar10-in-2-minutes-7b5aaecd9cdd
- https://www.coursera.org/specializations/tensorflow2-deeplearning

Real Time Applications in Day to Day life and to Industry

1. **Image Classification:** Image classification is a fundamental task in computer vision that involves training a model to recognize and categorize images into predefined classes or labels. TensorFlow provides a powerful platform for developing image classification models.

Salient Key Features:

1.Data Preparation

2.Pre-trained Models

3.Transfer Learning

4.Custom Model Development

5.Training and Optimization

6.Evaluation Metrics

7.Deployment

8.Applications

2. Speech Recognition: Speech recognition is a technology that converts spoken language into written text. TensorFlow provides powerful tools and models for developing speech recognition systems. Here are the key features:

1. Acoustic Modeling
2. Language Modeling
3. Automatic Speech Recognition (ASR)
4. Training Data Augmentation
5. Transfer Learning
6. Speech Enhancement
7. On-device Speech Recognition

3. Natural Language Processing (NLP) pertains to the domain of artificial intelligence that centers on the interaction between computers and human language. TensorFlow provides a diverse array of tools and models specifically designed to facilitate the development of applications related to NLP. The following are the principal characteristics:

1. Text Preprocessing
2. Word Embeddings
3. Sequence Modeling
4. Text Classification
5. Language Translation
6. Text Generation
7. Transfer Learning

PRESCRIBED TEXT BOOKS AND REFERENCE BOOKS

TEXT BOOKS:

1.Ben Auffarth, Generative AI with LangChain, Packt Publishing, 2023.

2.Amit Bahree, Generative AI in Action, Manning Publication, First Edition, 2023.

REFERENCES:

1.David Foster, Generative Deep Learning, 2nd Edition, O'Reilly Media, 2023.

2.Numa Dhamani and Maggie Engler, Introduction to Generative AI, Manning Publication, First Edition, 2024.

3.Valentina Alto, Modern Generative AI with ChatGPT and OpenAI Models, Packt publications, 2024.

MINI PROJECT SUGGESTIONS

- **TEAM 1:** Detecting Spam using TensorFlow
- **TEAM 2:** Image Classification with TensorFlow
- **TEAM3:** Optical Character Recognition using TensorFlow
- **TEAM 4:** Sudoku Solver using TensorFlow
- **TEAM 5:** Optical Character Recognition using TensorFlow

TOUGH QUESTIONS SIMILAR TO GATE

1. In a VAE, the Evidence Lower Bound (ELBO) is given by:

$$\mathcal{L}(\theta,\phi)=\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]-\mathrm{KL}(q_\phi(z|x)\|p(z))$$

What happens if the KL divergence term dominates the reconstruction loss?

A) The VAE generates diverse but noisy samples.

B) The VAE ignores latent structure, behaving like a plain autoencoder.

C) The VAE collapses to the prior, ignoring input data.

D) The VAE overfits the training data.

Ans: C

2. For two Gaussians

$p=\mathcal{N}(\mu_1,\sigma_1^2)$ and $q=\mathcal{N}(\mu_2,\sigma_2^2)$, the KL divergence $\mathrm{KL}(p\|q)$ is:

A) $\log\frac{\sigma_2}{\sigma_1}+\frac{\sigma_1^2+(\mu_1-\mu_2)^2}{2\sigma_2^2}-\frac{1}{2}$

B) $\log\frac{\sigma_1}{\sigma_2}+\frac{\sigma_1^2+(\mu_1-\mu_2)^2}{2\sigma_2^2}-\frac{1}{2}$

C) $\log\frac{\sigma_2}{\sigma_1}+\frac{\sigma_2^2+(\mu_1-\mu_2)^2}{2\sigma_1^2}-\frac{1}{2}$

D) $\log\frac{\sigma_1}{\sigma_2}+\frac{\sigma_2^2+(\mu_1-\mu_2)^2}{2\sigma_1^2}-\frac{1}{2}$

Ans: C

TOUGH QUESTIONS SIMILAR TO GATE

3. How does IAF differ from MAF (Masked Autoregressive Flow)?

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\phi(z|x) \| p(z))$$

A) IAF is faster for sampling but slower for density estimation.

B) IAF is faster for density estimation but slower for sampling.

C) IAF cannot model multimodal distributions.

D) IAF requires invertible neural networks.

Ans: A

4. For the reparameterization trick is $z \sim \mathcal{N}(\mu, \sigma^2)$,

A) $z = \mu + \sigma \cdot \epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$.

B) $z = \mu + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

C) $z = \sigma + \mu \cdot \epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$.

D) $z = \mu \cdot \sigma \cdot \epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$.

Ans: A

TOUGH QUESTIONS SIMILAR TO GATE

5. Which code correctly loads CIFAR-10 and normalizes pixels to

[−1,1][−1,1]?

A) (x_train, y_train), _ = tf.keras.datasets.cifar10.load_data()

```
x_train = x_train / 255.0
```

B) (x_train, y_train), _ = tf.keras.datasets.cifar10.load_data()

```
x_train = (x_train / 127.5) - 1.0
```

C) (x_train, y_train), _ = tf.keras.datasets.cifar10.load_data()

```
x_train = (x_train / 127.5) - 1.0
```

D) (x_train, y_train), _ = tf.keras.datasets.cifar10.load_data()

```
x_train = tf.image.per_image_standardization(x_train) .
```

Ans: B

6. Which code defines a VAE encoder with stochastic sampling?

A) class Encoder(tf.keras.layers.Layer):

```
def call(self, x):
z_mean = dense(x)
z_log_var = dense(x)
return z_mean, z_log_var
```

TOUGH QUESTIONS SIMILAR TO GATE

B) class Encoder(tf.keras.layers.Layer):

```
    def call(self, x):
    z_mean = dense(x)
     z_log_var = dense(x)
     epsilon = tf.random.normal(tf.shape(z_mean))
     return z_mean + tf.exp(z_log_var) * epsilon
```

C) class Encoder(tf.keras.layers.Layer):

```
     def call(self, x):
      z_mean = dense(x)
      z_log_var = dense(x)
      epsilon = tf.random.normal(tf.shape(z_mean))
      return z_mean + tf.exp(0.5 * z_log_var) * epsilon
```

D) class Encoder(tf.keras.layers.Layer):

```
     def call(self, x):
      z = dense(x)
      return tf.random.normal(z)
```

Ans : C

TOUGH QUESTIONS SIMILAR TO GATE

7. The reconstruction loss for a VAE with binary cross-entropy is suitable for:

A) Grayscale images

B) RGB images normalized to [0, 1]

C) RGB images normalized to [-1, 1]

D) Text data

Ans : B

8. For a flow $z = f(x)$, the probability $p(x)$ is computed as:

A) $p(x) = p(z) \cdot \det \left| \frac{\partial f}{\partial x} \right|$

B) $p(x) = p(z) \cdot \left| \frac{\partial f}{\partial x} \right|^{-1}$

C) $p(x) = p(z) \cdot \det \left| \frac{\partial f}{\partial x} \right|^{-1}$

D) $p(x) = p(z) + \log \left| \frac{\partial f}{\partial x} \right|$

Ans : B

TOUGH QUESTIONS SIMILAR TO GATE

9. Your VAE generates blurry images. What is the least likely fix?

A) Increase the latent space dimension.

B) Replace MSE loss with SSIM loss.

C) Use a stronger decoder.

D) Remove the KL term.

Ans : D

10. How does IAF improve upon RealNVP?

A) IAF scales to higher dimensions better.

B) IAF avoids masking entirely.

C) IAF uses autoregressive transforms in reverse.

D) IAF eliminates the need for Jacobian calculations.

Ans : C

THANK YOU