Advanced GenAI Fundamentals & LLMs

A language model (LM) is a probabilistic model of text
I wrote to the zoo to send me a pet. They sent me a

Word	...	lion	elephant	dog	cat	panther	alligator	...
Probability		0.1	0.1	0.3	0.2	0.05	0.02	

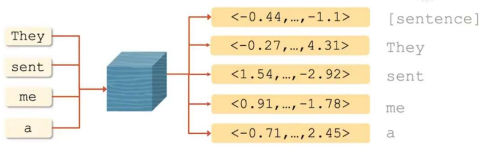
GenAI (Generative AI)
Meaning:
GenAI woh AI hota hai jo **naya content generate** karta hai. Naya matlab literally naya — sirf copy-paste ya search nahi.
Kya generate kar sakta hai?
- Text (blogs, emails, code)
- Images (logos, posters)
- Audio (music, voice)
- Video
- Code

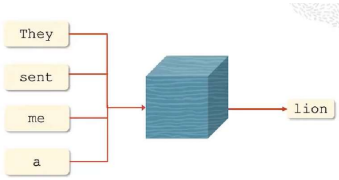
Hard truth:
GenAI koi magic nahi hai. Ye **patterns seekh ke output nikalta hai**. Agar data ghatiya hai → output bhi ghatiya hoga.
Example samajho:
- Prompt diya: *"Ek startup pitch likh"*
  → AI khud se structure, words, flow bana ke likhe → **GenAI**

LLM (Large Language Model)
Meaning:
LLM ek **specific type ka GenAI model** hota hai jo **language** pe focus karta hai.
⚠ Important: **Har LLM GenAI hota hai, lekin har GenAI LLM nahi hota.**
LLM kya karta hai?
- Text samajhta hai
- Text likhta hai
- Questions answer karta hai
- Code likhta / explain karta hai
- Translation, summarization

LLM ka kaam ka basic logic:
- Ye words ko "samajhta" nahi
- Ye **next word predict** karta hai based on probability
  Bas itna hi. Intelligence illusion hai, magic nahi.

Encoders and Decoders



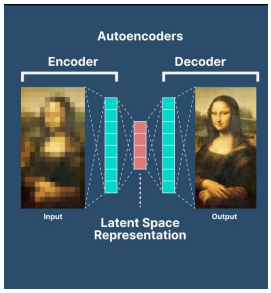


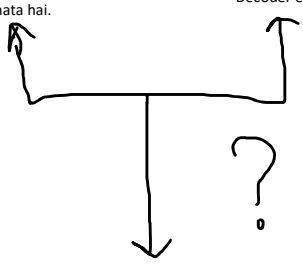
Encoder - models that convert a sequence of words to an embedding (vector representation)
Examples: MiniLM, Embed-light, BERT, ROBERTA, DistillBERT, SBERT,...

Socho encoder ek **student** hai jo lecture sun ke **notes** banata hai.

Decoder - models take a sequence of words and output next word
Examples - GPT-4, Llama, BLOOM, Falcon, ...

Decoder ek **teacher** hai jo notes dekh ke **answer likhta** hai.



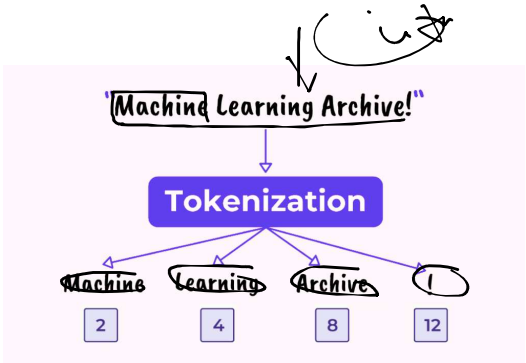


- Encoder **samajhta** hai
- Decoder **express** karta hai

Akele encoder bas samajh lega, bolega nahi
Akele decoder bolega, par samjhega nahi
Isliye **pair mandatory** hai for generation tasks.

```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Load dataset
(x_train, _), (x_test, _) = tf.keras.datasets.mnist.load_data()

# Normalize
x_train = x_train / 255.0
x_test = x_test / 255.0

# Flatten images
x_train = x_train.reshape(-1, 784)
x_test = x_test.reshape(-1, 784)

# Encoder
encoder = models.Sequential([
   layers.Dense(128, activation='relu'),
   layers.Dense(32, activation='relu') ])

# Decoder
decoder = models.Sequential([
   layers.Dense(128, activation='relu'),
   layers.Dense(784, activation='sigmoid')
])

# Autoencoder = Encoder + Decoder
autoencoder = models.Sequential([encoder, decoder])

autoencoder.compile(optimizer='adam', loss='mse')

# Train
autoencoder.fit(
   x_train, x_train,
   epochs=10,
   batch_size=256,
   validation_data=(x_test, x_test)
)

# Test reconstruction
decoded_imgs = autoencoder.predict(x_test[:10])

# Plot original vs reconstructed
plt.figure(figsize=(10,4))
for i in range(10):
   plt.subplot(2,10,i+1)
   plt.imshow(x_test[i].reshape(28,28), cmap='gray')
   plt.axis('off')

   plt.subplot(2,10,i+11)
   plt.imshow(decoded_imgs[i].reshape(28,28), cmap='gray')
   plt.axis('off')

plt.show()
```

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers

# Simple dataset
input_texts = ["hello", "world", "skill", "train"]
target_texts = [text[::-1] for text in input_texts]

# Character set
chars = sorted(list(set("".join(input_texts))))
char_index = {c:i+1 for i,c in enumerate(chars)}

def encode(text):
   return [char_index[c] for c in text]

max_len = max(len(t) for t in input_texts)

X = tf.keras.preprocessing.sequence.pad_sequences(
   [encode(t) for t in input_texts], maxlen=max_len
)
```




```
from transformers import AutoTokenizer
```

Tokenization — Text → IDs
Role: Pre-processing
- Human language → tokens
- Tokens → token IDs (numbers)

Example:

"I love AI"
→ ["I", " love", " AI"]
→ [101, 234, 876]

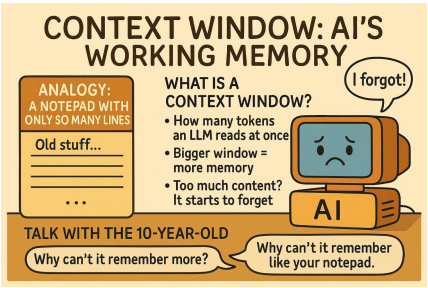
Use hota hai:
- Har NLP model me
- Encoder ho ya Decoder ho — dono ke pehle

⚠ Tokenization bina model ke bhi hoti hai
☞ Model ka part **nahi**, entry gate hai

```
max_len = max(len(t) for t in input_texts)

X = tf.keras.preprocessing.sequence.pad_sequences(
    [encode(t) for t in input_texts], maxlen=max_len
)

y = tf.keras.preprocessing.sequence.pad_sequences(
    [encode(t[::-1]) for t in input_texts], maxlen=max_len
)

# Model
encoder_input = layers.Input(shape=(max_len,))
embed = layers.Embedding(len(char_index)+1, 8)(encoder_input)
encoded = layers.LSTM(32)(embed)

decoder_input = layers.RepeatVector(max_len)(encoded)
decoded = layers.LSTM(32, return_sequences=True)(decoder_input)
output = layers.TimeDistributed(
    layers.Dense(len(char_index)+1, activation='softmax')
)(decoded)

model = tf.keras.Model(encoder_input, output)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy')

model.fit(X, y[..., None], epochs=300, verbose=0)

# Test
pred = model.predict(X)
print("Input:", input_texts)
print("Predicted:", [
    "".join(chars[np.argmax(c)-1] for c in word if np.argmax(c) != 0)
    for word in pred
])
```

Context Window — **LLM ki RAM**

Context window = max tokens jo model ek time me dekh sakta hai
(prompt + history + documents + response)
Agar limit cross hui → **purana data drop**.

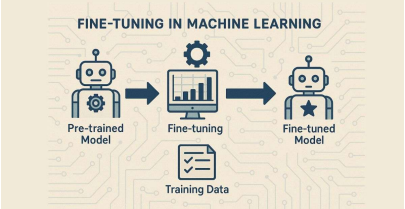


⚠ Tokenization bina model ke bhi hoti hai
👉 Model ka part **nahi**, entry gate hai

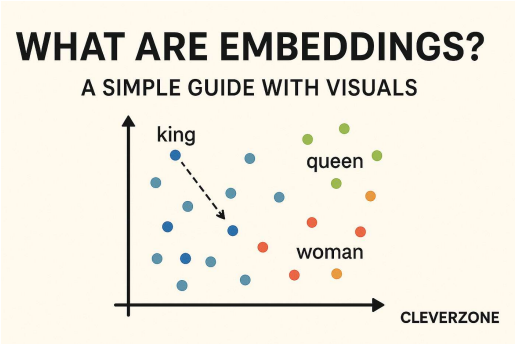
```
from transformers import AutoTokenizer

# Pretrained tokenizer (BERT)
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")

text = "I love Artificial Intelligence"

# Tokenization
tokens = tokenizer.tokenize(text)
token_ids = tokenizer.convert_tokens_to_ids(tokens)

print("Original Text:", text)
print("Tokens:", tokens)
print("Token IDs:", token_ids)
```

Fine-tuning — **Behavior training, NOT knowledge**

Sabse bada myth
❌ Fine-tuning se model ko naya knowledge yaad kara denge
Ye galat hai. **100% galat.**



Task	Encoder	Decoder
Translation	English samajhta	Hindi likhta
Chatbot	User message samajhta	Reply generate
Speech → Text	Audio samajhta	Words likhta
Image Captioning	Image samajhta	Caption likhta



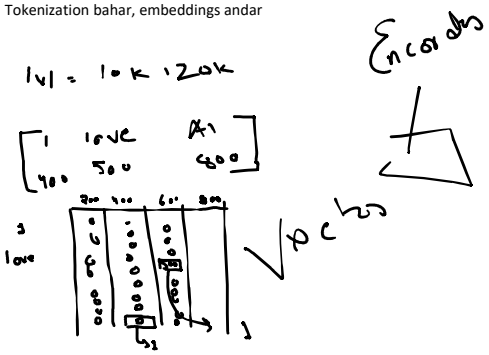
Embeddings — IDs → Vectors
Role: Representation
- Token IDs → dense vectors
- Meaning encode hota hai

Example:

~~"I" → [0.12, 0.98, 1.45, ...]~~

Use hota hai:
- Encoder models (BERT)
- Decoder models (GPT)
- RAG, Search, Recommendation

⚠ Embeddings **model ke andar hoti hain**
Tokenization bahar, embeddings andar



```
from torch.nn.functional import cosine_similarity

text1 = "I love AI"
text2 = "I enjoy artificial intelligence"

emb1 = model(**tokenizer(text1,
return_tensors="pt")).last_hidden_state.mean(dim=1)
emb2 = model(**tokenizer(text2,
return_tensors="pt")).last_hidden_state.mean(dim=1)

similarity = cosine_similarity(emb1, emb2)
```

```
print("Similarity score:", similarity.item())
```