Technical AI terms:

Large Language Models (LLMs): These are AI models specifically designed to understand and generate human language by being trained on a vast amount of text data.

Variational Autoencoders (VAEs): A type of AI model that can be used to create new images. It has two main parts: the encoder reduces data to a simpler form, and the decoder expands it back to generate new content.

Latent Space: A compressed representation of data that the autoencoder creates in a simpler, smaller form, which captures the most important features needed to reconstruct or generate new data.

Parameters: Parameters are the variables that the model learns during training. They are internal to the model and are adjusted through the learning process. In the context of neural networks, parameters typically include weights and biases.

Weights: Weights are coefficients for the input data. They are used in calculations to determine the importance or influence of input variables on the model's output. In a neural network, each connection between neurons has an associated weight.

Biases (not mentioned in the video): Biases are additional constants attached to neurons and are added to the weighted input before the activation function is applied. Biases ensure that even when all the inputs are zero, there can still be a non-zero output.

Hyperparameters: Hyperparameters, unlike parameters, are not learned from the data. They are more like settings or configurations for the learning process. They are set prior to the training process and remain constant during training. They are external to the model and are used to control the learning process.

Autoregressive text generation: Autoregressive text generation is like a game where the computer guesses the next word in a sentence based on the words that came before it. It keeps doing this to make full sentences.

Latent space decoding: Imagine if you had a map of all the possible images you could create, with each point on the map being a different image. Latent space decoding is like picking a point on that map and bringing the image at that point to life.

Diffusion models: Diffusion models start with a picture that's full of random dots like TV static, and then they slowly clean it up, adding bits of the actual picture until it looks just like a real photo or painting.

Generative Adversarial Networks (GANs): A system where two neural networks, one to generate data and one to judge it, work against each other. This competition helps improve the quality of the generated results.

Recurrent Neural Networks (RNNs): A network that's really good at handling sequences, like sentences or melodies, because it processes one piece at a time and remembers what it saw before.

Transformer-based models: A more advanced type that looks at whole sequences at once, not one piece at a time, making it faster and smarter at tasks like writing sentences or translating languages.

Sequential Data: Data that is connected in a specific order, like words in a sentence or steps in a dance routine.

Deepfakes: Highly realistic fake videos or images created by AI, which can make it seem like people are saying or doing things they never actually did.

Automation: The use of technology to perform tasks without human intervention, which can increase efficiency but also may replace jobs done by people.

Copyright Issues: Legal problems that arise when someone uses work without permission, potentially impacting the original creator's rights.

Carbon Footprint: The total amount of greenhouse gases produced directly or indirectly by activities or entities, like running large-scale AI models.

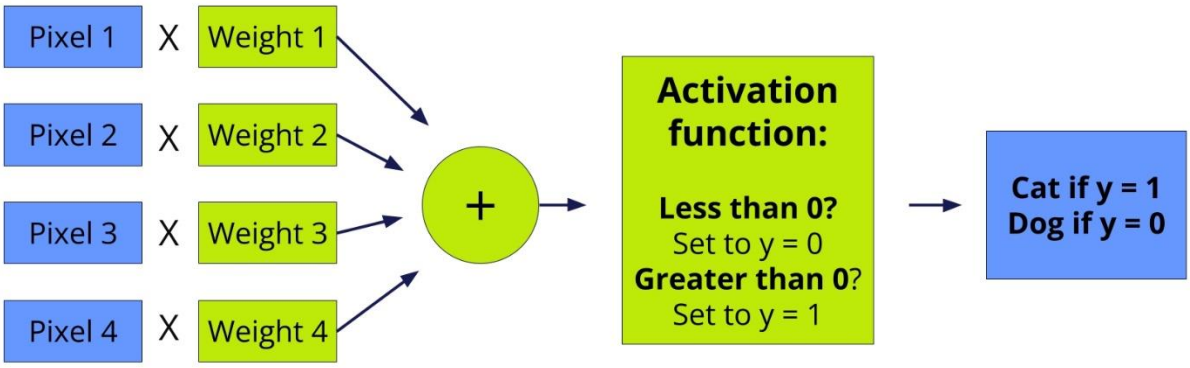
A perceptron is an essential component in the world of AI, acting as a binary classifier capable of deciding whether data, like an image, belongs to one class or another. It works by adjusting its weighted inputs—think of these like dials fine-tuning a radio signal—until it becomes better at predicting the right class for the data. This process is known as learning, and it shows us that even complex tasks start with small, simple steps.

Perceptron: A basic computational model in machine learning that makes decisions by weighing input data. It's like a mini-decision maker that labels data as one thing or another.

Binary Classifier: A type of system that categorizes data into one of two groups. Picture a light switch that can be flipped to either on or off.

Vector of Numbers: A sequence of numbers arranged in order, which together represent one piece of data.

Activation Function: A mathematical equation that decides whether the perceptron's calculated sum from the inputs is enough to trigger a positive or negative output.



...

Activation Functions:

https://www.geeksforgeeks.org/activation-functions-neural-networks/

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

The multi-layer perceptron is a powerful tool in the world of machine learning, capable of making smart decisions by mimicking the way our brain's neurons work. This amazing system can learn from its experiences, growing smarter over time as it processes information through layers, and eventually, it can predict answers with astonishing accuracy!

Multi-Layer Perceptron (MLP): A type of artificial neural network that has multiple layers of nodes, each layer learning to recognize increasingly complex features of the input data.

Input Layer: The first layer in an MLP where the raw data is initially received.

Output Layer: The last layer in an MLP that produces the final result or prediction of the network.

Hidden Layers: Layers between the input and output that perform complex data transformations.

We learned that training deep neural networks involves guided adjustments to improve their performance on tasks like image recognition. By gradually refining the network's parameters and learning from mistakes, these networks become smarter and more skilled at predicting outcomes. The marvel of this technology is its ability to turn raw data into meaningful insights.

Labeled Dataset: This is a collection of data where each piece of information comes with a correct answer or label. It's like a quiz with the questions and answers already provided.

Gradient Descent: This method helps find the best settings for a neural network by slowly tweaking them to reduce errors, similar to finding the lowest point in a valley.

Cost Function: Imagine it as a score that tells you how wrong your network's predictions are. The goal is to make this score as low as possible.

Learning Rate: This hyperparameter specifies how big the steps are when adjusting the neural network's settings during training. Too big, and you might skip over the best setting; too small, and it'll take a very long time to get there.

Backpropagation: Short for backward propagation of errors. This is like a feedback system that tells each part of the neural network how much it contributed to any mistakes, so it can learn and do better next time.

A **hold-out set** is a subset of the dataset that is set aside and not used during model training. It is used to evaluate the model’s performance on unseen data.

How It Works:

1. **Dataset Splitting** – The original dataset is divided into:
    - **Training Set**: Used to train the model.
    - **Validation Set** (optional): Used for hyperparameter tuning and model selection.
    - **Hold-Out (Test) Set**: Used only for final evaluation after training is complete.
2. **Purpose of the Hold-Out Set**:
    - Provides an unbiased estimate of model performance.
    - Helps detect overfitting, ensuring that the model generalizes well to new data.
    - Used in benchmarking different models.
3. **Typical Splitting Ratios**:
    - **Training (70-80%)**, **Validation (10-15%)**, **Hold-Out/Test (10-15%)**.
    - If no validation set is used, the split is usually **Train (80%) / Hold-Out (20%)**.
4. **Difference Between Validation and Hold-Out Set**:
    - **Validation Set**: Used to fine-tune model hyperparameters.
    - **Hold-Out (Test) Set**: Used only once after model tuning for final evaluation.

This method is essential in machine learning to ensure that models perform well on real-world data.

Labeled Dataset: This is a collection of data where each piece of information comes with a correct answer or label. It's like a quiz with the questions and answers already provided.

Gradient Descent: This method helps find the best settings for a neural network by slowly tweaking them to reduce errors, similar to finding the lowest point in a valley.

Cost Function: Imagine it as a score that tells you how wrong your network's predictions are. The goal is to make this score as low as possible.

Learning Rate: This hyperparameter specifies how big the steps are when adjusting the neural network's settings during training. Too big, and you might skip over the best setting; too small, and it'll take a very long time to get there.

Backpropagation: Short for backward propagation of errors. This is like a feedback system that tells each part of the neural network how much it contributed to any mistakes, so it can learn and do better next time.

PyTorch is a dynamic and powerful tool for building and training machine learning models. It simplifies the process with its fundamental building blocks like tensors and neural networks and offers effective ways to define objectives and improve models using loss functions and optimizers. By leveraging PyTorch, anyone can gain the skills to work with large amounts of data and develop cutting-edge AI applications.

https://pytorch.org/

PyTorch tensors are crucial tools in the world of programming and data science, which work somewhat like building blocks helping to shape and manage data effortlessly. These tensors allow us to deal with data in multiple dimensions, which is especially handy when working with things like images or more complex structures. Getting to know tensors is a step forward in understanding how PyTorch simplifies the processes of deep learning, enabling us to perform intricate numerical computations efficiently.

Tensors: Generalized versions of vectors and matrices that can have any number of dimensions (i.e. multi-dimensional arrays). They hold data for processing with operations like addition or multiplication.

Matrix operations: Calculations involving matrices, which are two-dimensional arrays, like adding two matrices together or multiplying them.

Scalar values: Single numbers or quantities that only have magnitude, not direction (for example, the number 7 or 3.14).

Linear algebra: An area of mathematics focusing on vector spaces and operations that can be performed on vectors and matrices.

PyTorch loss functions are essential tools that help in improving the accuracy of a model by measuring errors. These functions come in different forms to tackle various problems, like deciding between categories (classification) or predicting values (regression). Understanding and using these functions correctly is key to making smart, effective models that do a great job at the tasks they're designed for!

Loss functions: They measure how well a model is performing by calculating the difference between the model's predictions and the actual results.

Cross entropy loss: This is a measure used when a model needs to choose between categories (like whether an image shows a cat or a dog), and it shows how well the model's predictions align with the actual categories.

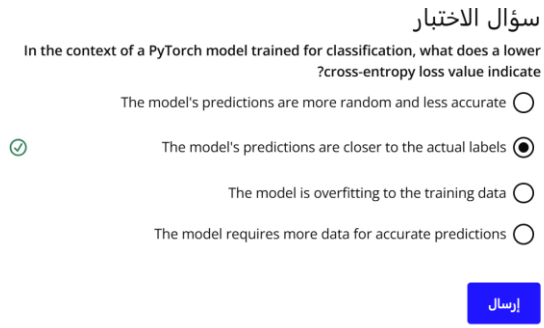
Mean squared error: This shows the average of the squares of the differences between predicted numbers (like a predicted price) and the actual numbers. It's often used for predicting continuous values rather than categories.

PyTorch loss functions are essential tools that help in improving the accuracy of a model by measuring errors. These functions come in different forms to tackle various problems, like deciding between categories (classification) or predicting values (regression). Understanding and using these functions correctly is key to making smart, effective models that do a great job at the tasks they're designed for!

Loss functions: They measure how well a model is performing by calculating the difference between the model's predictions and the actual results.

Cross entropy loss: This is a measure used when a model needs to choose between categories (like whether an image shows a cat or a dog), and it shows how well the model's predictions align with the actual categories.

Mean squared error: This shows the average of the squares of the differences between predicted numbers (like a predicted price) and the actual numbers. It's often used for predicting continuous values rather than categories.

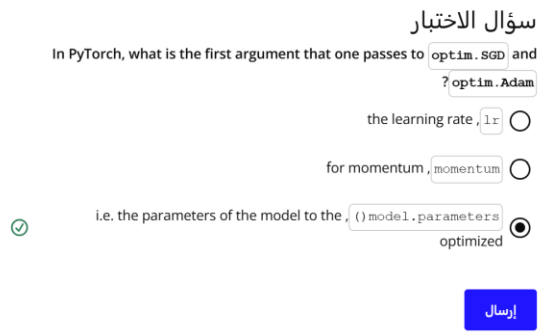


PyTorch optimizers are important tools that help improve how a neural network learns from data by adjusting the model's parameters. By using these optimizers, like stochastic gradient descent (SGD) with momentum or Adam, we can quickly get started learning!

Gradients: Directions and amounts by which a function increases most. The parameters can be changed in a direction opposite to the gradient of the loss function in order to reduce the loss.

Learning Rate: This hyperparameter specifies how big the steps are when adjusting the neural network's settings during training. Too big, and you might skip over the best setting; too small, and it'll take a very long time to get there.

Momentum: A technique that helps accelerate the optimizer in the right direction and dampens oscillations.

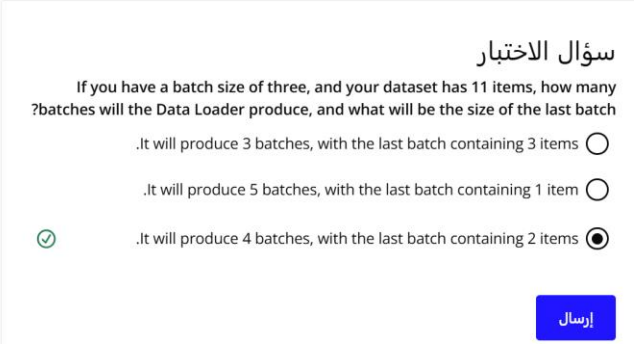


PyTorch Dataset class: This is like a recipe that tells your computer how to get the data it needs to learn from, including where to find it and how to parse it, if necessary.

PyTorch Data Loader: Think of this as a delivery truck that brings the data to your AI in small, manageable loads called batches; this makes it easier for the AI to process and learn from the data.

Batches: Batches are small, evenly divided parts of data that the AI looks at and learns from each step of the way.

Shuffle: It means mixing up the data so that it's not in the same order every time, which helps the AI learn better.



Correct! Since 11 divided by 3 results in 3 batches with 2 left over, there will be an additional batch for the remaining items.

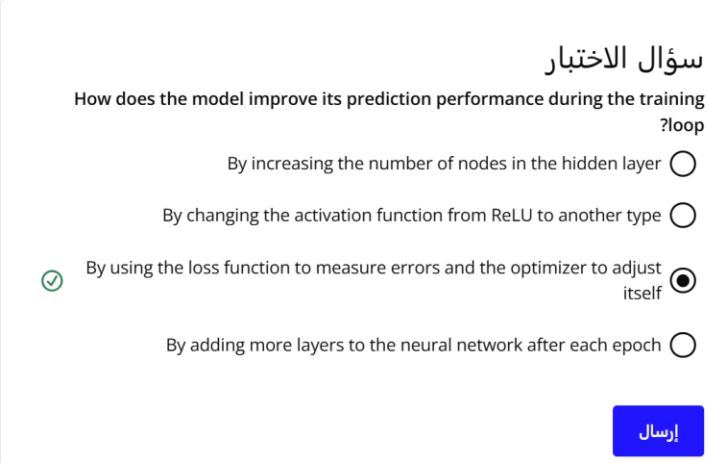
Training Loop: The cycle that a neural network goes through many times to learn from the data by making predictions, checking errors, and improving itself.

Batches: Batches are small, evenly divided parts of data that the AI looks at and learns from each step of the way.

Epochs: A complete pass through the entire training dataset. The more epochs, the more the computer goes over the material to learn.

Loss functions: They measure how well a model is performing by calculating the difference between the model's predictions and the actual results.

Optimizer: Part of the neural network's brain that makes decisions on how to change the network to get better at its job.

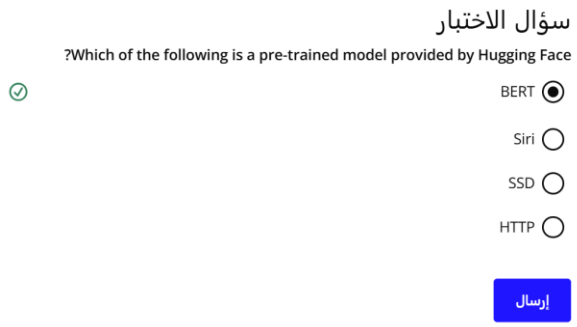


Tokenizers: These work like a translator, converting the words we use into smaller parts and creating a secret code that computers can understand and work with.

Models: These are like the brain for computers, allowing them to learn and make decisions based on information they've been fed.

Datasets: Think of datasets as textbooks for computer models. They are collections of information that models study to learn and improve.

Trainers: Trainers are the coaches for computer models. They help these models get better at their tasks by practicing and providing guidance. HuggingFace Trainers implement the PyTorch training loop for you, so you can focus instead on other aspects of working on the model.

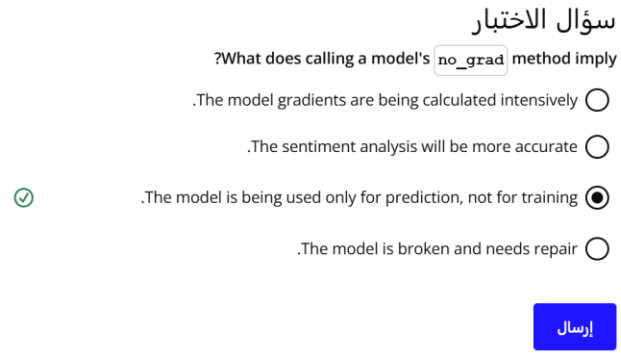


Tokenization: It's like cutting a sentence into individual pieces, such as words or characters, to make it easier to analyze.

Tokens: These are the pieces you get after cutting up text during tokenization, kind of like individual Lego blocks that can be words, parts of words, or even single letters. These tokens are converted to numerical values for models to understand.

Pre-trained Model: This is a ready-made model that has been previously taught with a lot of data.

Uncased: This means that the model treats uppercase and lowercase letters as the same.



IMDb dataset: A dataset of movie reviews that can be used to train a machine learning model to understand human sentiments.

Apache Arrow: A software framework that allows for fast data processing

Hugging Face trainers offer a simplified approach to training generative AI models, making it easier to set up and run complex machine learning tasks. This tool wraps up the hard parts, like handling data and carrying out the training process, allowing us to focus on the big picture and achieve better outcomes with our AI endeavors.

Truncating: This refers to shortening longer pieces of text to fit a certain size limit.

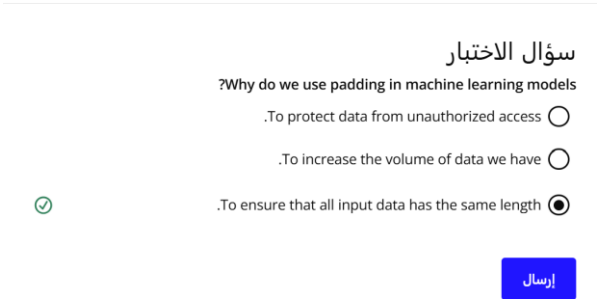
Padding: Adding extra data to shorter texts to reach a uniform length for processing.

Batches: Batches are small, evenly divided parts of data that the AI looks at and learns from each step of the way.

Batch Size: The number of data samples that the machine considers in one go during training.

Epochs: A complete pass through the entire training dataset. The more epochs, the more the computer goes over the material to learn.

Dataset Splits: Dividing the dataset into parts for different uses, such as training the model and testing how well it works.



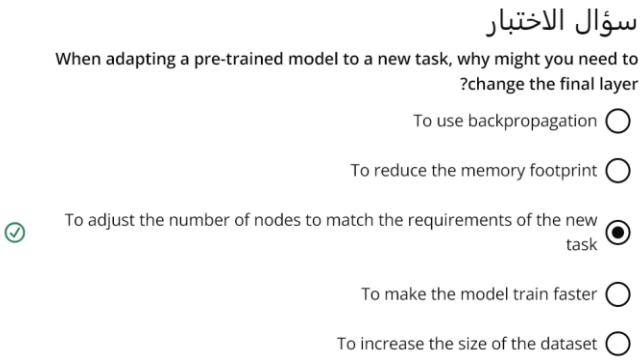
Correct - Padding is used to standardize data length.

By using pre-trained models and the magic of transfer learning, the hard work of training an AI model from zero can be bypassed, making it easier and quicker to get the job done. By leveraging the knowledge a model distills from a large dataset, we can reduce the amount of training needed to get a performant model.

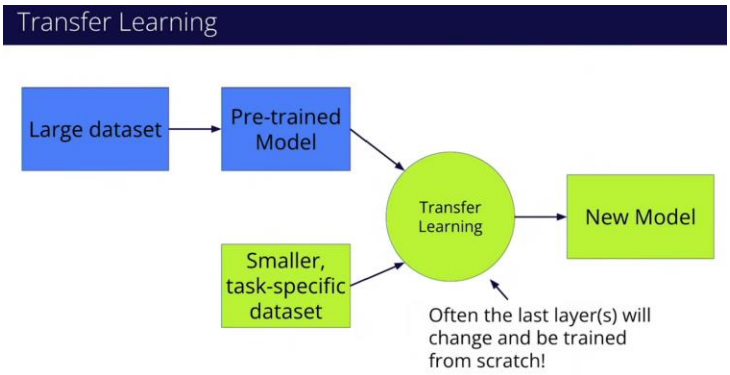
- If we wanted to create a plant identification app for mobile devices, we might use **MobileNetV3(opens in a new tab)** and train it on a dataset containing photos of different plant species.
- If we wanted to create a social networking spam classifier, we might use BERT and train it on a dataset containing samples of spam and not-spam text.

Technical Terms Explained:

Transfer learning: The process where knowledge from a pre-trained model is applied to a new, but related task.



Correct! The number of nodes at the output of the network might need to change.



A foundation model is a powerful AI tool that can do many different things after being trained on lots of diverse data. These models are incredibly versatile and provide a solid base for creating various AI applications, like a strong foundation holds up different kind of buildings. By using a foundation model, we have a strong starting point for building specialized AI tasks.

Foundation Model: A large AI model trained on a wide variety of data, which can do many tasks without much extra training.

Adapted: Modified or adjusted to suit new conditions or a new purpose, i.e. in the context of foundation models.

Generalize: The ability of a model to apply what it has learned from its training data to new, unseen data.

Foundation Models and Traditional Models are two distinct approaches in the field of artificial intelligence with different strengths. Foundation Models, which are built on large, diverse datasets, have the incredible ability to adapt and perform well on many different tasks. In contrast, Traditional Models specialize in specific tasks by learning from smaller,

focused datasets, making them more straightforward and efficient for targeted applications.

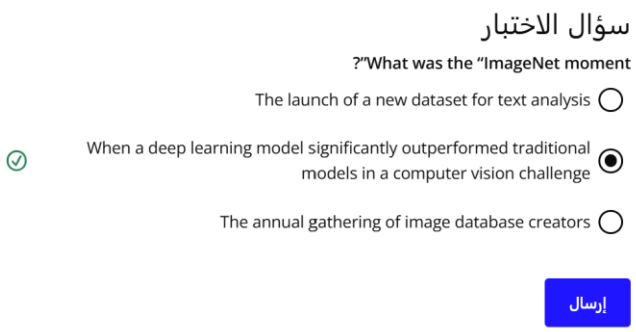
Sequential data: Information that is arranged in a specific order, such as words in a sentence or events in time.

Self-attention mechanism: The self-attention mechanism in a transformer is a process where each element in a sequence computes its representation by attending to and weighing the importance of all elements in the sequence, allowing the model to capture complex relationships and dependencies.

Benchmarks matter because they are the standards that help us measure and accelerate progress in AI. They offer a common ground for comparing different AI models and encouraging innovation, providing important stepping stones on the path to more advanced AI technologies.

Robustness: The strength of an AI model to maintain its performance despite challenges or changes in data.

Open Access: Making data sets freely available to the public, so that anyone can use them for research and develop AI technologies.



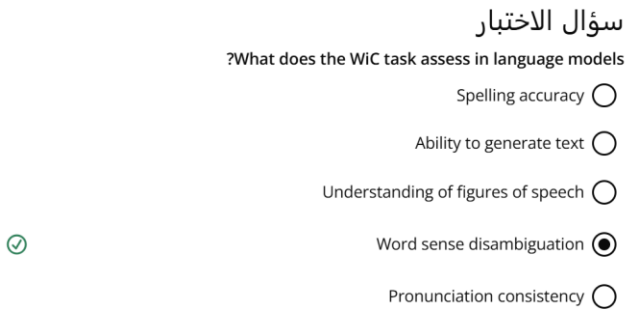
Correct as the ImageNet moment refers to the point when deep learning models began to excel in visual recognition tasks.

The GLUE benchmarks serve as an essential tool to assess an AI's grasp of human language, covering diverse tasks, from grammar checking to complex sentence relationship analysis. By putting AI models through these varied linguistic challenges, we can gauge their readiness for real-world tasks and uncover any potential weaknesses.

Semantic Equivalence: When different phrases or sentences convey the same meaning or idea.

Textual Entailment: The relationship between text fragments where one fragment follows logically from the other.

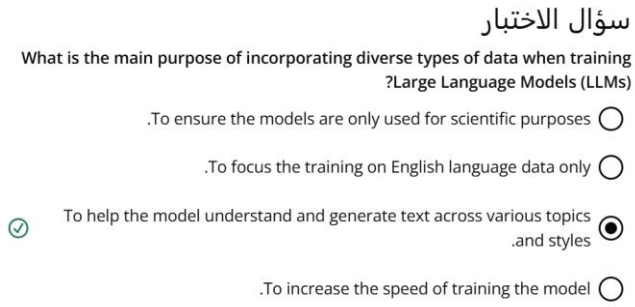
SuperGlue is designed as a successor to the original GLUE benchmark. It's a more advanced benchmark aimed at presenting even more challenging language understanding tasks for AI models. Created to push the boundaries of what AI can understand and process in natural language, SuperGlue emerged as models began to achieve human parity on the GLUE benchmark. It also features a public leaderboard, facilitating the direct comparison of models and enabling the tracking of progress over time.



Correct, WiC evaluates if a model can tell whether a word is used with the same meaning in two different sentences

Preprocessing: This is the process of preparing and cleaning data before it is used to train a machine learning model. It might involve removing errors, irrelevant information, or formatting the data in a way that the model can easily learn from it.

Fine-tuning: After a model has been pre-trained on a large dataset, fine-tuning is an additional training step where the model is further refined with specific data to improve its performance on a particular type of task.

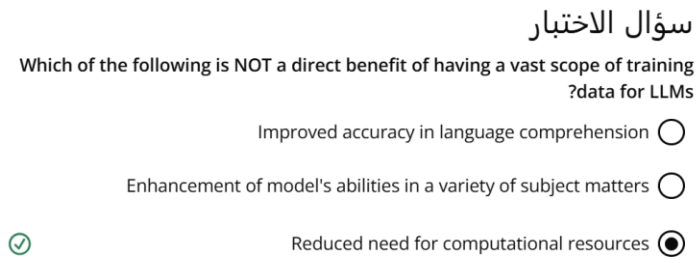


Correct, a wider array of data helps LLMs become more versatile and capable of managing different types of language and content.

The scale of data for Large Language Models (LLMs) is tremendously vast, involving datasets that could equate to millions of books. The sheer size is pivotal for the model's understanding and mastery of language through exposure to diverse words and structures.

Gigabytes/Terabytes: Units of digital information storage. One gigabyte (GB) is about 1 billion bytes, and one terabyte (TB) is about 1,000 gigabytes. In terms of text, a single gigabyte can hold roughly 1,000 books.

Common Crawl: An open repository of web crawl data. Essentially, it is a large collection of content from the internet that is gathered by automatically scraping the web.



Correct, vast scope of training data does not directly reduce computational resources, but instead may require more to process and understand the language better.

Biases in training data deeply influence the outcomes of AI models, reflecting societal issues that require attention. Ways to approach this challenge include promoting diversity in development teams, seeking diverse data sources, and ensuring continued vigilance through bias detection and model monitoring.

Selection Bias: When the data used to train an AI model does not accurately represent the whole population or situation by virtue of the selection process, e.g. those choosing the data will tend to choose dataset their are aware of

Historical Bias: Prejudices and societal inequalities of the past that are reflected in the data, influencing the AI in a way that perpetuates these outdated beliefs.

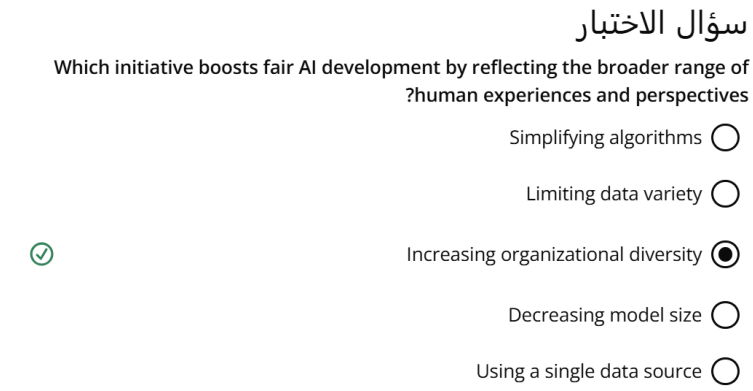
Confirmation Bias: The tendency to favor information that confirms pre-existing beliefs, which can affect what data is selected for AI training.

Discriminatory Outcomes: Unfair results produced by AI that disadvantage certain groups, often due to biases in the training data or malicious actors.

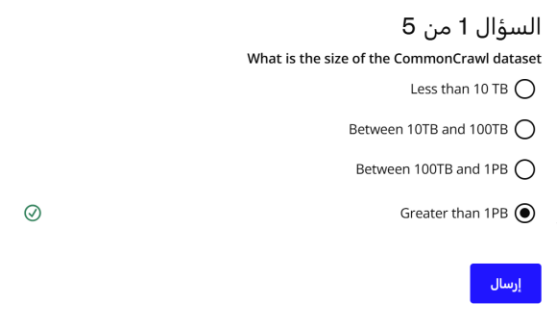
Echo Chambers: Situations where biased AI reinforces and amplifies existing biases, leading to a narrow and distorted sphere of information.

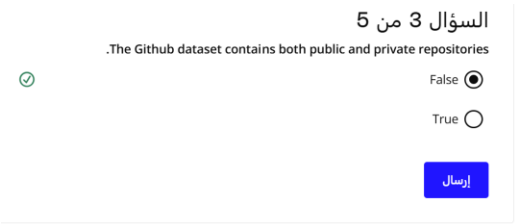
Bias Detection and Correction: Processes and algorithms designed to identify and remove biases from data before it's used to train AI models.

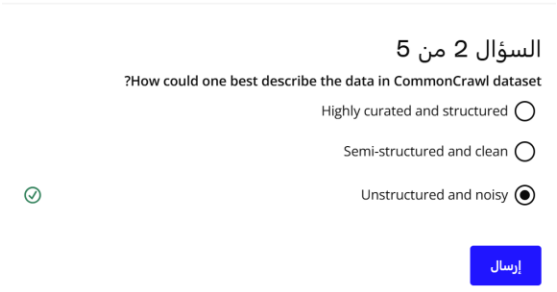
Transparency and Accountability: Openness about how AI models are trained and the nature of their data, ensuring that developers are answerable for their AI's performance and impact.

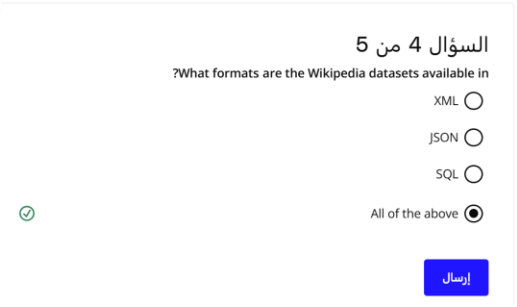


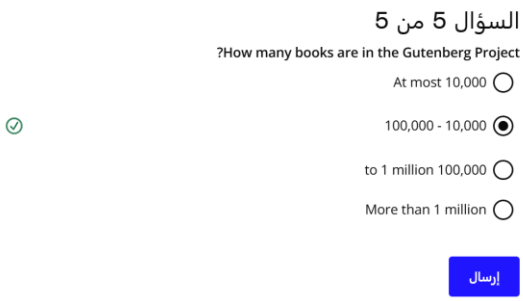
Correct: A diverse team brings different perspectives to AI development, ultimately aiding in mitigating biases.





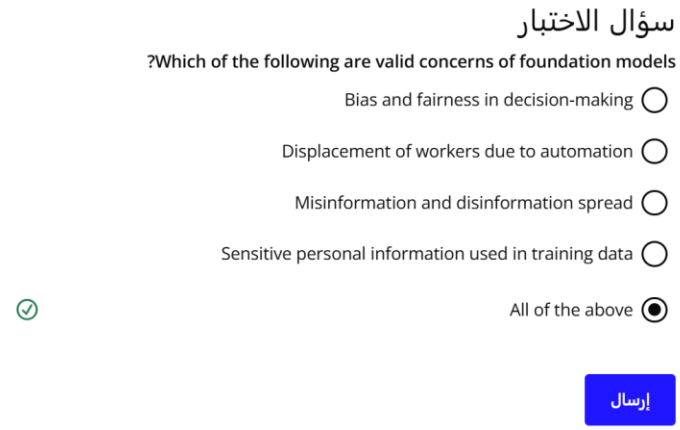






Synthetic Voices: These are computer-generated voices that are often indistinguishable from real human voices. AI models have been trained on samples of speech to produce these realistic voice outputs.

Content Provenance Tools: Tools designed to track the origin and history of digital content. They help verify the authenticity of the content by providing information about its creation, modification, and distribution history.



Adaptation in AI is a crucial step to enhance the capabilities of foundation models, allowing them to cater to specific tasks and domains. This process is about tailoring pre-trained AI systems with new data, ensuring they perform optimally in specialized applications and respect privacy constraints. Reaping the benefits of adaptation leads to AI models that are not only versatile but also more aligned with the unique needs of organizations and industries.

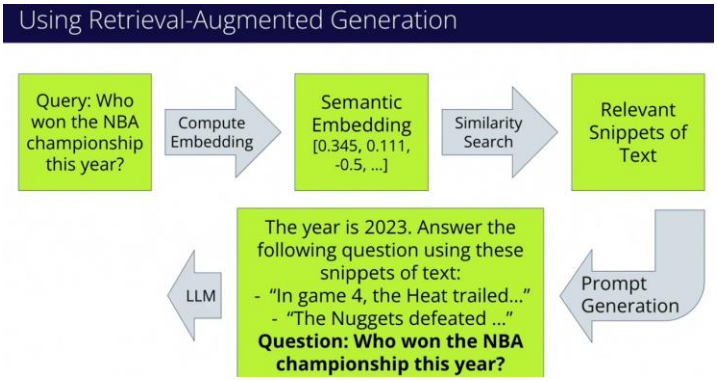
Fine-tuning: This is a technique in machine learning where an already trained model is further trained (or tuned) on a new, typically smaller, dataset for better performance on a specific task.

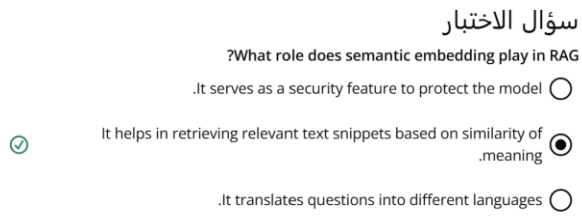
Retrieval-Augmented Generation (RAG) is a powerful approach for keeping Generative AI models informed with the most recent data, particularly when dealing with domain-specific questions. It cleverly combines the comprehensive understanding capacity of a large language model (LLM) with the most up-to-date information pulled from a database of relevant text snippets. The beauty of this system is in its ability to ensure that responses remain accurate and reflective of the latest developments.

Semantic-embedding: A representation of text in a high-dimensional space where distances between points correspond to semantic similarity. Phrases with similar meanings are closer together.

Cosine similarity: A metric used to measure how similar two vectors are, typically used in the context of semantic embeddings to assess similarity of meanings.

Vector databases: Specialized databases designed to store and handle vector data, often employed for facilitating fast and efficient similarity searches.





Correct! Semantic embedding is used to find text snippets that are semantically similar to the question, aiding in the retrieval process.

Prompt Design Techniques are innovative strategies for tailoring AI foundation models to specific tasks, fostering better performance in various domains. These methods enable us to guide the AI's output by carefully constructing the prompts we provide, enhancing the model's relevance and efficiency in generating responses.

Domain-Specific Task: A task that is specialized or relevant to a particular area of knowledge or industry, often requiring tailored AI responses.

Prompt tuning is a technique in generative AI which allows models to target specific tasks effectively. By crafting prompts, whether through a hands-on approach with hard prompts or through an automated process with soft prompts, we enhance the model's predictive capabilities.

Prompt: In AI, a prompt is an input given to the model to generate a specific response or output.

Prompt Tuning: This is a method to improve AI models by optimizing prompts so that the model produces better results for specific tasks.

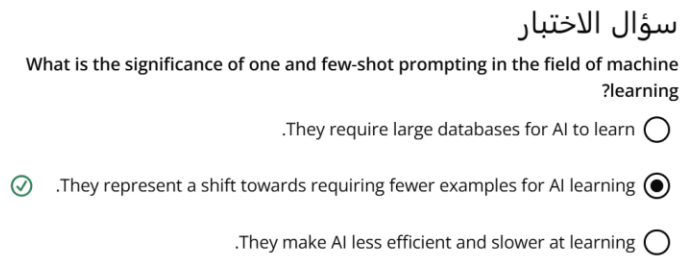
Hard Prompt: A manually created template used to guide an AI model's predictions. It requires human ingenuity to craft effective prompts.

Soft Prompt: A series of tokens or embeddings optimized through deep learning to help guide model predictions, without necessarily making sense to humans.

One and few-shot prompting represent cutting-edge techniques that enable AI to adapt and perform tasks with minimal instructions. Instead of relying on extensive databases for learning, these methods guide generative AI through just one or a few examples, streamlining the learning process and demonstrating its ability to generalize solutions to new problems. This innovative approach marks a significant advancement in machine learning, empowering AI to quickly adjust to specialized tasks and showcasing the incredible potential for efficiency in teaching AI new concepts.

One-shot prompting: Giving an AI model a single example to learn from before it attempts a similar task.

Few-shot prompting: Providing an AI model with a small set of examples, such as five or fewer, from which it can learn to generalize and perform tasks.



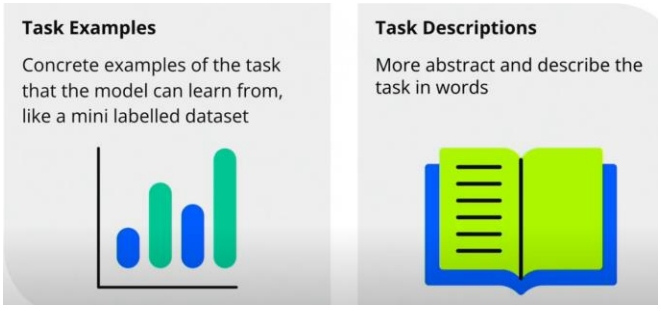
Correct: These methods reduce the number of examples needed to teach AI.

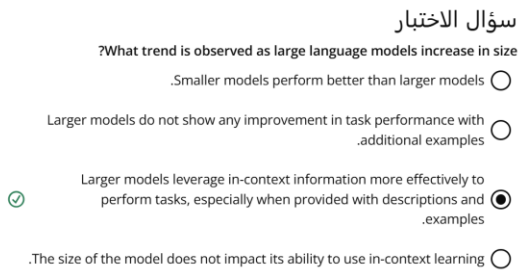
Zero-shot prompting is a remarkable technique where a generative AI model can take on new tasks without the need for specific training examples. This process leverages the AI's extensive pre-existing knowledge gained from learning patterns across vast datasets. It empowers the AI to infer and generalize effectively to provide answers and solutions in contexts that were not expressly covered during its initial training.

Zero-shot prompting: This refers to the capability of an AI model to correctly respond to a prompt or question it hasn't explicitly been trained to answer, relying solely on its prior knowledge and training.

When performing few-shot, one-shot, or zero-shot learning, we can pass information to the model within the prompt in the form of examples, descriptions, or other data. When we rely on a model using information from within the prompt itself instead of relying on what is stored within its own parameters we are using *in-context learning*.

As these AI models grow in size, their ability to absorb and use in-context information significantly improves, showcasing their potential to adapt to various tasks effectively. The progress in this field is inspiring, as these advances hint at an exciting future where such models could be even more intuitive and useful.

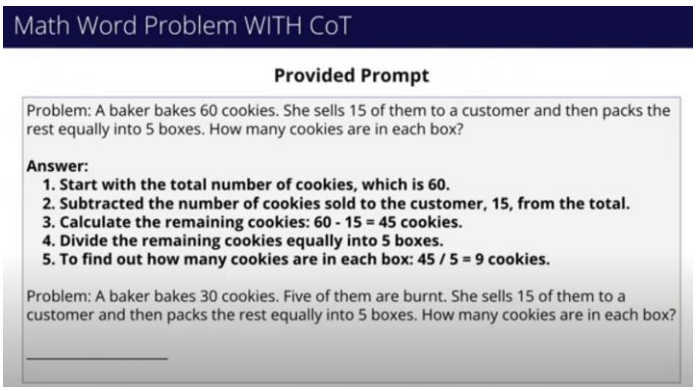


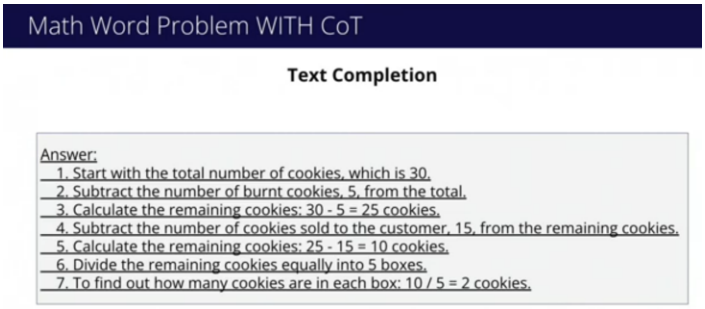


Correct: The video talks about larger models in blue performing better and using examples and descriptions more effectively.

Chain-of-Thought Prompting is a vital technique for enhancing the reasoning capabilities of large language models by breaking down complex problems into intermediate steps that lead to a solution. By providing models with a line of reasoning, they can more effectively tackle problems that require more advanced problem-solving processes, enabling them to deduce information, such as the number of cookies in a box, after considering all variables.

Chain-of-Thought Prompting: A method of guiding a language model through a step-by-step reasoning process to help it solve complex tasks by explicitly detailing the logic needed to reach a conclusion.





السؤال 1 من 4

:This prompt is an example of

one-shot prompting ◯

two-shot prompting ◯

three-shot prompting ◉

Yes! Since there are three labelled examples this is three-short prompting.

Using probing to train a classifier is a powerful approach to tailor generative AI foundation models, like BERT, for specific applications. By adding a modestly-sized neural network, known as a classification head, to a foundation model, one can specialize in particular tasks such as sentiment analysis. This technique involves freezing the original model's parameters and only adjusting the classification head through training with labeled data. Ultimately, this process simplifies adapting sophisticated AI systems to our needs, providing a practical tool for developing efficient and targeted machine learning solutions.

Probing: This is a method of examining what information is contained in different parts of a machine learning model.

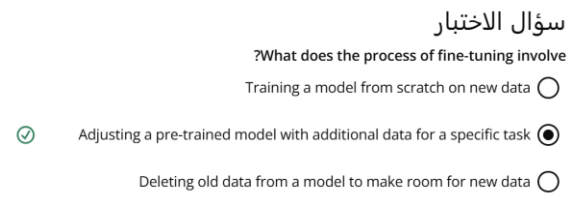
Linear Probing: A simple form of probing that involves attaching a linear classifier to a pre-trained model to adapt it to a new task without modifying the original model.

Classification Head: It is the part of a neural network that is tailored to classify input data into defined categories.

Fine-tuning is an important phase in enhancing the abilities of generative AI models, making them adept at specific tasks. By introducing additional data to these powerful models, they can be tailored to meet particular requirements, which is invaluable in making AI more effective and efficient. Although this process comes with its challenges, such as the need for significant computational resources and data, the outcome is a more specialized and capable AI system that can bring value to a wide range of applications.

Fine-tuning: This is the process of adjusting a pre-trained model so it performs better on a new, similar task. It's like teaching an experienced doctor a new medical procedure; they're already a doctor, but they're improving their skills in a particular area.

Catastrophic Forgetting: This happens when a model learns something new but forgets what it learned before. Imagine if you crammed for a history test and did great, but then forgot most of what you learned when you started studying for a math test.



Correct! Fine-tuning updates a model that has already learned general information with new data, to specialize it.

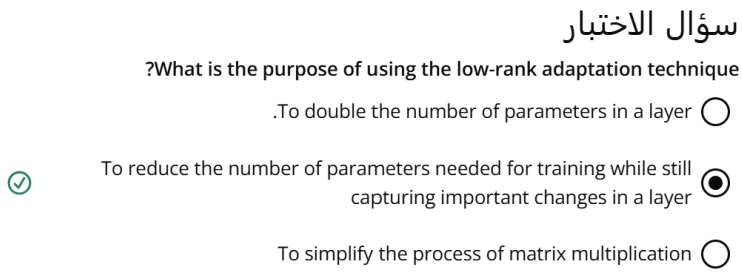
Parameter-efficient fine-tuning (PEFT) is a technique crucial for adapting large language models more efficiently, with the bonus of not requiring heavy computational power. This approach includes various strategies to update only a small set of parameters, thereby maintaining a balance between model adaptability and resource consumption. The techniques ensure that models can be swiftly deployed in different industrial contexts, considering both time constraints and the necessity for scaling operations efficiently.

Parameter-efficient fine-tuning: A method of updating a predefined subset of a model's parameters to tailor it to specific tasks, without the need to modify the entire model, thus saving computational resources.

Frozen Parameters: In the context of machine learning, this refers to model parameters that are not changed or updated during the process of training or fine-tuning.

Low-Rank Adaptation (LoRA): A technique where a large matrix is approximated using two smaller matrices, greatly reducing the number of parameters that need to be trained during fine-tuning.

Adapters: Additional model components inserted at various layers; only the parameters of these adapters are trained, not of the entire model.



Correct! Low rank adaptation uses smaller matrices to approximate important changes in a layer, reducing the number of trainable parameters without significant loss of function.

In the dynamic field of Artificial Intelligence, the shift from building models from the ground up to adapting existing foundational models is becoming increasingly prevalent. Mastery of adaptation techniques—from prompting methods such as few-shot learning and chain-of-thought prompting, to parameter-efficient fine-tuning techniques such as low-rank adaptation—empowers us to leverage pre-existing powerful models for diverse applications, enhancing creativity and efficiency in our projects.