PPT 2:

Hough Transform. Used to connect disjoint edge points
Works in mc plane instead of xy plane.

A point can have infinite lines passing through it. Thus we convert our single point in the xy plane to a line in the cm plane.

So if A and B are two points connected by a line in the spatial domain, they will be intersecting lines in the Hough space.

PPT 6:

Brainstorming of Inception Module:

The Inception module is part of a deep learning architecture called InceptionNet. In the original version of the Inception module, the use of larger filters like 5x5 in a convolutional layer led to a significant increase in computational cost. For example, when processing an input image of size 32 x 32 x 200 using a 5x5 convolutional layer with 32 filters, the total number of multiplications needed to compute the result was over 163 million.

To address this issue, researchers at Google brainstormed and came up with a new version of the Inception module that reduced the computational cost by using smaller filters and pooling operations in parallel. This new version was able to achieve similar accuracy while requiring much less computation, making it more practical for use in real-world applications.

Dimensionality Reduction Layer (1x1 Conv Layer):

The dimensionality reduction layer is a layer in a neural network that helps to make computations faster and more efficient. It does this by reducing the number of channels in the input data without changing its spatial dimensions. For example, if the input data has 200 channels, the dimensionality reduction layer can reduce it to 16 channels using a 1x1 convolution. This smaller data is then used as input to the larger convolutional layers, making them faster and more efficient. This technique is used in the Inception module to reduce the computational cost of using larger kernels.

Does shrinking the depth representation size so dramatically hurt the performance of neural networks?

Adding a reduced layer before the 3 × 3 and 5 × 5 convolutional layers to reduce their computational cost can actually improve the performance of the neural network. By reducing the depth of the input feature maps, we can speed up the computation and prevent overfitting. This

is because fewer parameters mean less chance of overfitting and also because having too many parameters can slow down training due to the need for more computations. So, in moderation, shrinking the depth representation size can actually help the performance of the neural network.

This note refers to the Inception module and its modifications. In the original Inception module, there was a problem with the computational cost when using larger filters like 5x5. To address this issue, a 1x1 convolutional layer was added before the larger filters. This layer reduces the depth of the input feature map, which reduces the number of computations needed for the larger filters. Similarly, a 1x1 convolutional layer is added after the 3x3 max-pooling layer to reduce the depth of its output, which can help improve the efficiency of the network. These modifications help to reduce the computational cost of the Inception module without sacrificing performance.

GoogleNet/Inception V1:

The GoogleNet/Inception V1 is a deep neural network that has many layers, which means that it can learn more complex features from the data it's trained on. However, when a network is too deep, it can be hard to train because the gradient signals that are used to update the weights in the network become weaker as they pass through many layers.

Google researchers found that adding extra classifiers to the middle of the network can help address this issue. These classifiers are connected to the intermediate layers and encourage the network to learn more discriminative features in the lower layers. This also increases the gradient signal that gets propagated back during training, which helps to improve the performance of the network. In addition, these auxiliary classifiers provide additional regularization, which can help to prevent overfitting.

Auxiliary classifiers:

Auxiliary classifiers are additional classifier layers added to a neural network to help improve its performance. They are typically placed at intermediate layers within the network, and their purpose is to encourage the network to learn more discriminative features early on in the training process.

In the case of GoogleNet/Inception V1, auxiliary classifiers were added to the network to help address the challenge of effectively propagating gradients through all the layers. By adding classifiers at intermediate stages of the network, the model is forced to learn more meaningful representations early on in the training process, which in turn helps improve the overall performance of the network. Additionally, the auxiliary classifiers provide an extra source of regularization that can help prevent overfitting.

Vanishing gradient:

In deep neural networks, during the training process, the gradient (i.e., the derivative of the loss function with respect to the network parameters) is propagated back from the output layer to the input layer to update the parameters using an optimization algorithm such as stochastic gradient descent. However, in very deep networks, the gradient may become very small (i.e., close to zero) as it propagates backward through the layers. This problem is known as the vanishing gradient problem.

The vanishing gradient problem occurs because the gradients of the activation functions used in the network become smaller as we move towards the input layer. This is because the gradients of many activation functions are less than one, and when we multiply these gradients across many layers during backpropagation, the overall gradient becomes smaller and smaller.

When the gradient becomes too small, the network parameters are updated very slowly, and the network may get stuck in a suboptimal solution or may not learn at all. There are several techniques used to mitigate the vanishing gradient problem, such as using activation functions with larger gradients, initializing the network parameters carefully, using skip connections, and using batch normalization.

ResNet:

Skip Connections:

Skip connections are a type of connection used in neural networks to allow the flow of information to bypass one or more layers in a network. In a traditional feedforward neural network, information flows from input to output layer through a series of intermediate layers. However, sometimes, during training, the gradients can become very small due to the vanishing gradient problem. This can cause the network to become stuck and make it difficult to learn from new training examples.

Skip connections were introduced as a solution to this problem. These connections allow information to bypass one or more layers in a network, thereby allowing gradients to flow directly from the output layer to earlier layers, avoiding the vanishing gradient problem. These connections are typically added by concatenating the output of one layer to the input of another layer deeper in the network, which preserves the spatial resolution of the input feature maps.

Skip connections have been used with great success in deep neural networks, particularly in convolutional neural networks (CNNs), where they have been shown to improve accuracy and training speed. One famous example of the use of skip connections is in the ResNet architecture, which achieved state-of-the-art results on image recognition tasks by using hundreds of layers with skip connections.

In neural networks, skip connections are used to help address the issue of vanishing gradients. To implement a skip connection, the output of one layer in the network is combined with the input of another layer that is deeper in the network. This is typically done by concatenating the output of the first layer with the input of the second layer.

By doing this, the network can preserve the original spatial resolution of the input feature maps. This can be important for certain types of data, such as images or videos, where the spatial relationships between different parts of the input can be important for making accurate predictions.

Overall, skip connections can help improve the performance of deep neural networks by allowing gradients to flow more easily through the network, and by preserving important spatial information in the input data.

How are skip connections different from dropouts?

Skip connections and dropouts are both techniques used in neural networks to prevent overfitting and improve performance, but they work in different ways.

Skip connections allow information to bypass one or more layers in a network, allowing gradients to flow directly from the output layer to earlier layers. This helps to avoid the vanishing gradient problem, which can occur when gradients become too small and fail to update the weights of earlier layers effectively. By preserving the spatial resolution of the input feature maps, skip connections can also help to improve the accuracy of object detection and segmentation tasks.

Dropouts, on the other hand, work by randomly dropping out (i.e., set to zero) some of the neurons in a layer during training. This prevents the neurons from co-adapting too much and overfitting to the training data and encourages the network to learn more robust features that are useful for generalization to new data.

In summary, skip connections and dropouts are both techniques used to improve neural network performance and prevent overfitting, but they work in different ways. Skip connections allow gradients to flow more effectively and preserve spatial resolution, while dropouts encourage the network to learn more robust and generalizable features.

MobileNet: Google’s approach to lightweight models:

MobileNet is a type of neural network that is designed to be efficient and work well on devices like phones or other small devices. It uses a special type of convolutional layer called Depth Wise Separable Convolution, along with a reduced layer (1x1 convolution), to make the network lighter and faster. It also has two hyperparameters that can be adjusted to balance accuracy and speed.

Overall, MobileNet is a good option for applications that need to process images quickly and efficiently on small devices.

MobileNet Approach: Depthwise Separable Convolutions:

In simpler terms, regular convolutional layers apply the same filter to all the channels in an image. Depthwise convolution, on the other hand, applies a different filter to each channel of an image, one channel at a time. This means that the output of the depthwise convolutional layer has the same number of channels as the input. After applying the depthwise convolutional layer, we use a pointwise convolutional layer which is not a reduced layer, but an expanded layer. The pointwise convolutional layer takes the output of the depthwise convolutional layer and applies a filter to each pixel location across all channels, which expands the output channels to a desired number. This approach helps in building lightweight deep neural networks suitable for mobile and embedded vision applications.

MobileNet's Two Model Lightening Hyperparameters:

MobileNet is a type of neural network designed to be used in mobile devices or other systems with limited computational power. To make it even smaller and faster, MobileNet uses two hyperparameters: width multiplier and resolution multiplier.

The width multiplier, represented by the symbol α, is used to make the network thinner at each layer. For example, if α is set to 0.5, the number of input channels at a layer is reduced by half, and the same applies to the number of output channels. The value of α is usually between 0 and 1.

The resolution multiplier, represented by the symbol ρ, is used to reduce the size of the input image and the internal representation of each layer by the same multiplier. For example, if ρ is set to 0.5, the input image and the internal representation of each layer will be reduced by half. In practice, the input resolution is set to values such as 224, 192, 160 or 128, which implicitly set the value of ρ. The value of ρ is also usually between 0 and 1.

Explain efficient net to me in detail but in simpler terms but also highlight the keywords:

EfficientNet is a family of neural networks designed for image classification tasks, which were introduced by Google in 2019. The keyword in the EfficientNet design is "compound scaling", which refers to optimizing three dimensions of the neural network: depth, width, and resolution.

Depth scaling means adding more layers to the network, while width scaling means adding more channels to each layer. Resolution scaling means resizing the input image to a smaller size, which reduces the computational cost of processing each image.

EfficientNet applies compound scaling by optimizing these three dimensions jointly. The model starts with a baseline network that has a fixed depth, width, and resolution, and then scales up all three dimensions simultaneously. The scaling coefficients used for each dimension are chosen through an optimization process that balances the model's accuracy and computational cost.

EfficientNet also makes use of a technique called "swish activation", which is a more computationally efficient activation function than the commonly used ReLU function. Additionally, it uses a novel approach called "MBConv", which is a combination of depthwise and pointwise convolutions, that further optimizes the model's performance.

The EfficientNet family includes multiple models with varying levels of depth, width, and resolution, that are designed for different computational budgets and accuracy requirements. The models are named using a compound scaling coefficient (e.g. EfficientNet-B0, EfficientNet-B1, etc.), where larger coefficients indicate larger models with better accuracy but higher computational cost.

What is the difference between efficient net and mobile net:

EfficientNet and MobileNet are both models designed by Google for efficient deep learning on mobile devices and other resource-constrained settings. However, there are some key differences between the two models.

- Architecture: While MobileNet is based on depthwise separable convolutions, EfficientNet uses a combination of depthwise and regular convolutions, as well as squeeze-and-excitation blocks and other advanced techniques.
- Model size: EfficientNet is generally larger and more complex than MobileNet. While MobileNet was designed to be lightweight and efficient, EfficientNet aims to achieve state-of-the-art performance while still being relatively efficient.
- Performance: EfficientNet has shown better performance than MobileNet on many benchmarks, particularly when the models are scaled up to larger sizes.
- Flexibility: EfficientNet has a range of model sizes, from EfficientNet-B0 (small and fast) to EfficientNet-B7 (large and accurate), which allows users to choose the right size for their specific needs. MobileNet has fewer size options.

In summary, while both EfficientNet and MobileNet are designed for efficient deep learning, EfficientNet is generally larger and more complex and has shown better performance on many benchmarks. However, MobileNet may still be a good choice for some use cases that require a very lightweight model.

What do CNNs see?

CNNs, or Convolutional Neural Networks, are a type of machine learning algorithm that can recognize patterns in images. When a CNN analyzes an image, it looks for certain features and uses those to make a prediction about what the image contains.

Class Activation Maps are one way to visualize what parts of the image a CNN is looking at when it makes a prediction. The Naïve approach simply highlights the parts of the image that had the highest activation, but this can lead to false positives. The Restricted approach, on the other hand, uses a threshold to only show the parts of the image that were most important in making the prediction.

Gradient Class Activation Maps, or Grad-CAM, is a more appropriate approach for all kinds of tasks, such as image classification, image captioning, visual question answering, and image segmentation. It uses the gradients of the predicted class with respect to the final convolutional layer to generate a heatmap that shows which parts of the image were most important in making the prediction. This allows us to better understand what the CNN "sees" when it looks at an image.

Grad-CAMs: Binary Image Classification:

Grad-CAMs, or Gradient Class Activation Maps, is a technique used to identify the important regions of an image that contributed to the prediction made by a convolutional neural network (CNN). This technique works for a variety of tasks such as image classification, image captioning, visual question answering and image segmentation.

To create a Grad-CAM for a binary image classification task, first, we need to create a custom model using a deep learning framework like tf.keras. This model takes an image as input and gives the last convolution output and the classification probability score as output. The classification probability score is obtained using the sigmoid activation function.

Next, we calculate the gradient of the classification activation value with respect to the last convolution output. If the classification probability score is greater than or equal to 0.5, we take the average of the gradient matrix. Otherwise, if it is less than 0.5, we put a negative sign on the gradient matrix and then take its average.

We then project this averaged gradient on the last convolutional feature map to form an image importance gradient heatmap. We only want the positive impact of this gradient heatmap, so we pass it through a ReLU function. We then normalize the gradient heatmap and resize it to the size of the original image. Finally, we project the gradient heatmap on to the image to obtain the regions that contributed most to the classification.

PPT 7:

Visual Embeddings:

Visual embeddings are a way to represent images as numbers, so computers can understand and work with them. It's like translating a picture into a code that a computer can read. These numbers are called a "dense vector representation" and they help with tasks like classifying images.

The output of this process is a set of numbers that represent the image. These numbers can be used to compare images to each other by measuring the distance between their sets of numbers.

Embeddings: Formal Definition:

An embedding is a way to turn something into a set of numbers that can be put into a space where things that are similar to each other are close together. This set of numbers is usually smaller than the original thing. It works for any kind of information.

Reverse Image Search/Image Recommendation Approach:

The reverse image search or image recommendation approach works by first turning all the images into sets of numbers called "visual embeddings" using a computer program that has already been trained on lots of images. These sets of numbers are then saved in a database.

When someone wants to search for or recommend an image, their image is also turned into a set of numbers, and the computer program calculates how different or similar it is to all the other images in the database. The program then finds the five images that are most similar to the one searched for, and shows them to the user.

To see if this approach works well, people use tests like checking how often the recommended images are the ones that the user actually wanted.

One Shot Learning: Face Recognition:

One-shot learning is a problem in computer vision where a machine learning algorithm tries to recognize an object or a face with only one or a few examples. Most machine learning algorithms need lots of examples to learn, but one-shot learning tries to do it with only a few examples.

For example, recognizing faces of people we don't have many pictures of. Deep learning algorithms usually need a lot of pictures to recognize faces, but with one-shot learning, we can try to do it with only one or a few pictures of each person.

To verify if someone is who they say they are based on their face, we first save a set of numbers that represents their face in a database. Then, when someone claims to be that person and provides a picture of their face, we calculate another set of numbers that represents their face from that picture.

We then measure how different or similar these two sets of numbers are by calculating the "distance" between them using a math formula called L2. If the distance is less than a certain "threshold" number, we say that the person is verified to be who they claim to be. If the distance is greater than the threshold, then it's not the same person.

Face Recognition:

Face recognition is a computer technology that identifies human faces in digital images or videos. It involves several steps.

The first step is face detection, where a computer program looks for faces in an image or video. It does this by using special algorithms like Haar cascades or object detection models like MTCNN.

Once the face is detected, the program tries to align the face to make it easier to work with. This is called face alignment.

Next, the program extracts a set of numbers that represents the face called an "embedding". This is done by using a special neural network that has been trained to turn faces into numbers.

Finally, the program uses these embeddings to identify or verify a person's identity by comparing the embedding of the person in the image with the embeddings of known people in a database.

Siamese Network:

A Siamese network is a type of artificial neural network that works on two inputs at the same time, like two images of faces. The network has two branches that are exactly the same and share the same weights. The inputs are fed through each branch and the network computes a set of numbers for each input, which are called output vectors.

The idea is to make sure that the output vectors of similar inputs are also similar. This can be useful for tasks like face recognition, where we want to recognize if two images of faces belong to the same person or not. The network learns to compare the output vectors of the two inputs and decide if they are similar or not, based on the shared weights.

Siamese Network: L1/L2 Distance and Classification

Calculate the distance between the feature vectors

In a Siamese network, we use two branches to compute feature vectors for two input images. These feature vectors represent the important information in the images that we want to compare.

To compare these feature vectors, we calculate the distance between them using either L1 or L2 distance. L1 distance is the sum of the absolute differences between each element in the two vectors, while L2 distance is the square root of the sum of the squared differences.

Once we have the distance between the two feature vectors, we pass it to the classification layer. The classification layer takes this distance vector and uses it to decide if the two input images belong to the same class or not, like if they are both images of the same person's face or not.

Siamese Network: Custom Loss:

In a Siamese network, we want to compare the feature vectors of two input images to see if they belong to the same class or not. To do this, we use a custom loss function that helps the network learn to distinguish between similar and dissimilar pairs of images.

One type of custom loss function is called contrastive loss. It works by trying to pull the feature vectors of similar images closer together and push the feature vectors of dissimilar images farther apart. The loss function penalizes the network when the distance between the feature

vectors of similar images is too large or when the distance between the feature vectors of dissimilar images is too small.

Another type of custom loss function is called triplet loss. It works by comparing the distance between the feature vectors of three images: an anchor image, a positive image (belonging to the same class as the anchor), and a negative image (belonging to a different class). The loss function penalizes the network when the distance between the anchor and negative images is too small or when the distance between the anchor and positive images is too large.

Both contrastive and triplet loss functions help the Siamese network learn to distinguish between similar and dissimilar pairs of images by adjusting the feature vectors based on the desired outcome.

Important Aspects of Face Recognition Systems:

Fairness: Face recognition systems should not be biased against any particular group of people. They should be designed and tested to ensure they work equally well across all demographics.

Transparency: The workings of the face recognition system should be transparent to the users, including how it collects, stores, and uses data.

Accountability: The owners and operators of the face recognition system should be accountable for its performance, accuracy, and impact on individuals and society.

Non-discrimination: Face recognition systems should not discriminate against people based on their race, gender, age, or other characteristics.

Notice and Consent: People should be informed about the use of face recognition systems and give their consent before their data is collected, stored, or used.

Lawful Surveillance: The use of face recognition systems should comply with relevant laws and regulations governing surveillance and privacy, including obtaining necessary permissions and protecting individuals' rights.