Deep Learning

BCSE-332L

Module 6:

Advanced Neural Networks

Dr . Saurabh Agrawal

Faculty Id: 20165

School of Computer Science and Engineering

VIT, Vellore-632014

Tamil Nadu, India

Outline

- Transfer Learning
- Transfer Learning Models
- Generative Adversarial Network and their variants
- Region based CNN
- Fast RCNN
- You Only Look Once
- Single Shot Detector

Transfer Learning

- Transfer learning is a smart method in machine learning where a model uses knowledge from one task to help with a different, but related, task.
- Instead of learning from zero, the model uses what it already knows to solve new problems faster and better.
- This is especially helpful when there isn't much data available.
- Transfer learning is making a big impact in areas like understanding language and recognizing images.

Transfer Learning

❑What is Transfer Learning?

❑Transfer learning is a technique in machine learning where a model trained on one task is used as the starting point for a model on a second task.

❑This can be useful when the second task is similar to the first task, or when there is limited data available for the second task.

❑By using the learned features from the first task as a starting point, the model can learn more quickly and effectively on the second task.

❑This can also help to prevent overfitting, as the model will have already learned general features that are likely to be useful in the second task.

Transfer Learning

❑Why do we need Transfer Learning?

❑**Limited Data**: In many real-world scenarios, obtaining a large amount of labeled data for training a model from scratch can be difficult and expensive. Transfer learning allows us to leverage pre-trained models and their knowledge, reducing the need for vast amounts of data.

❑**Improved Performance**: By starting with a pre-trained model, which has already learned from a large dataset, we can achieve better performance on new tasks more quickly. This is especially useful in applications where accuracy and efficiency are crucial.

❑**Time and Cost Efficiency**: Transfer learning saves time and resources because it speeds up the training process. Instead of training a new model from scratch, we can build on existing models and fine-tune them for specific tasks.

❑**Adaptability**: Models trained on one task can be adapted to perform well on related tasks. This adaptability makes transfer learning suitable for a wide range of applications, from image recognition to natural language processing.

Transfer Learning

❑**How does Transfer Learning work?:** This is a general summary of how transfer learning works:

❑**Pre-trained Model:** Start with a model that has previously been trained for a certain task using a large set of data. Frequently trained on extensive datasets, this model has identified general features and patterns relevant to numerous related jobs.

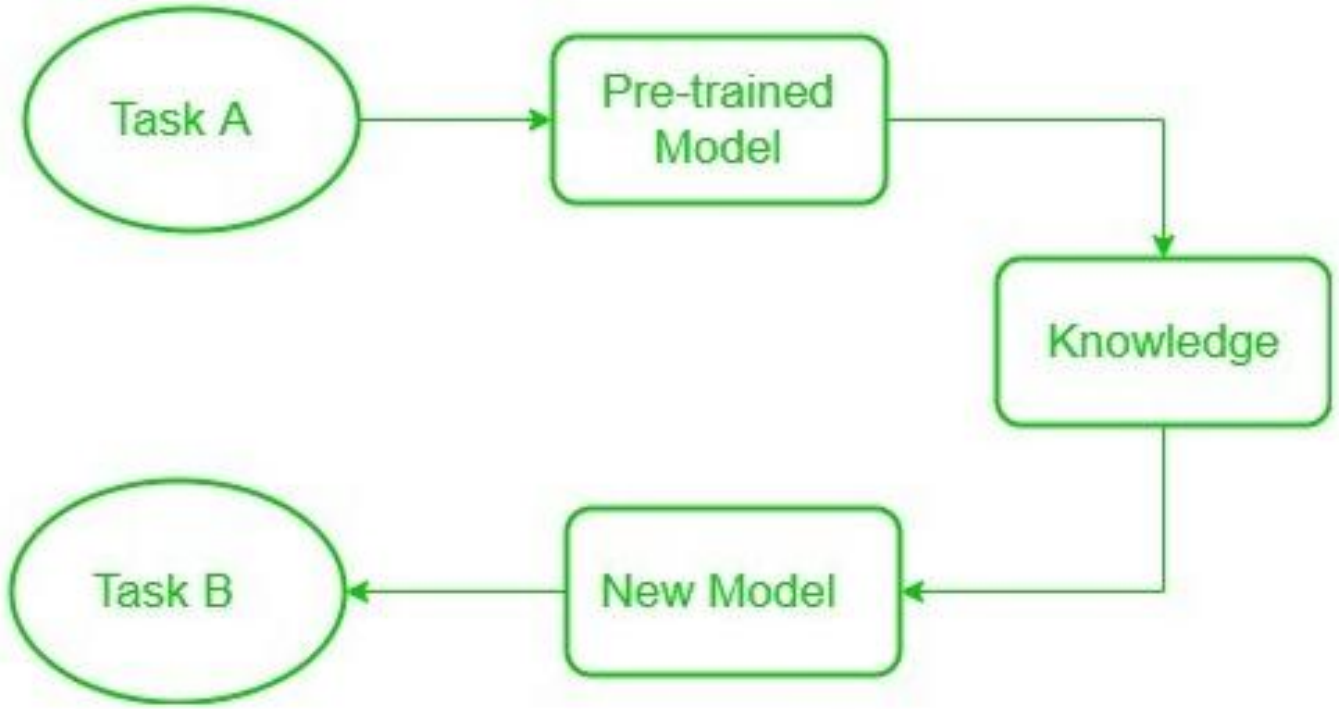
❑**Base Model:** The model that has been pre-trained is known as the base model. It is made up of layers that have utilized the incoming data to learn hierarchical feature representations.

❑**Transfer Layers:** In the pre-trained model, find a set of layers that capture generic information relevant to the new task as well as the previous one. Because they are prone to learning low-level information, these layers are frequently found near the top of the network.

❑**Fine-tuning:** Using the dataset from the new challenge to retrain the chosen layers. We define this procedure as fine-tuning. The goal is to preserve the knowledge from the pre-training while enabling the model to modify its parameters to better suit the demands of the current assignment.

Transfer Learning

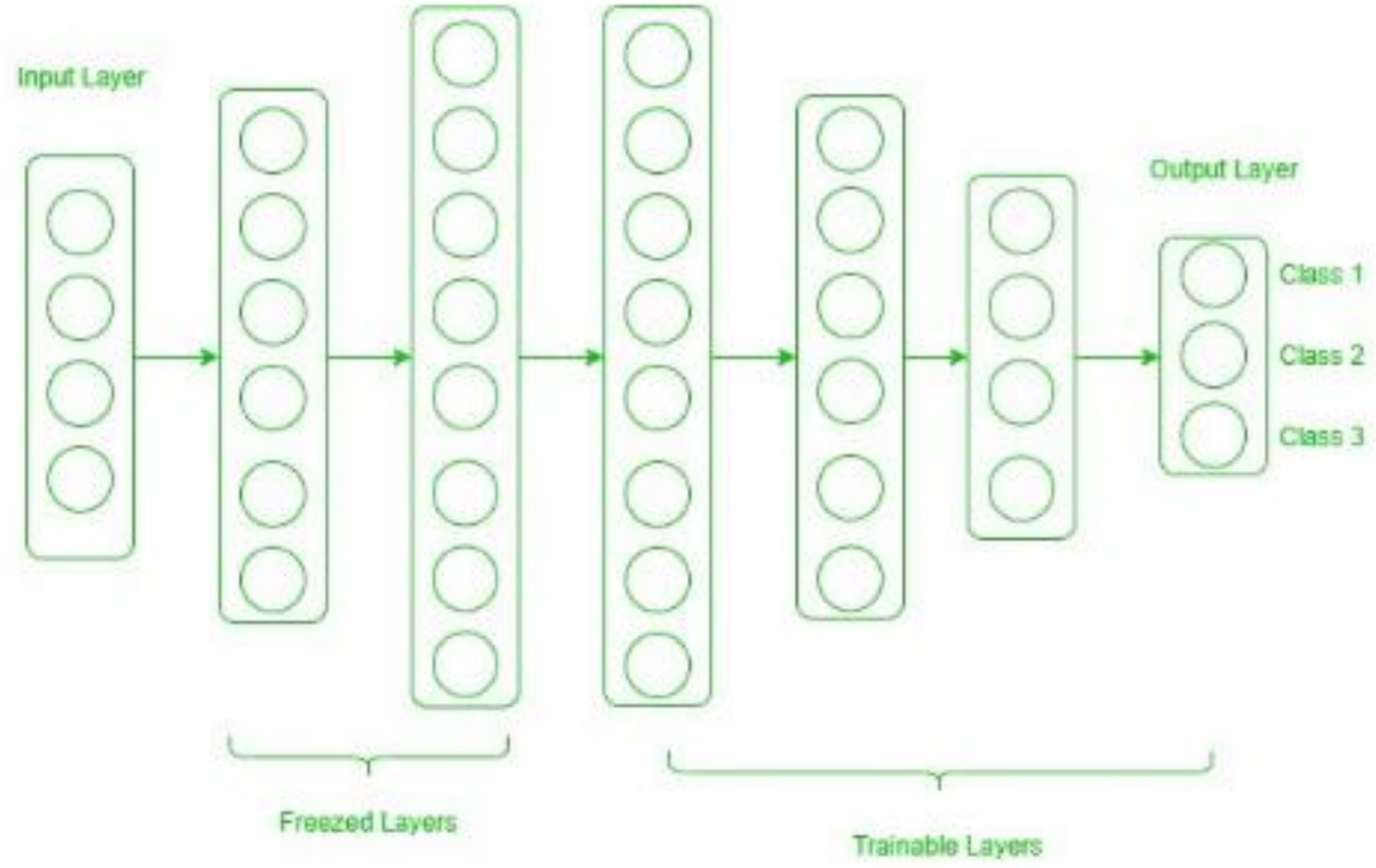
❑How does Transfer Learning work?



❑Low-level features learned for task A should be beneficial for learning of model for task B.

Transfer Learning

❑How does Transfer Learning work?: Freezed and Trainable Layers:



Transfer Learning

❑**How does Transfer Learning work?: Freezed and Trainable Layers:**

❑In transfer learning, there are two main components: frozen layers and modifiable layers.

1. **Frozen Layers**: These are the layers of a pre-trained model that are kept unchanged during the fine-tuning process. Frozen layers retain the knowledge learned from the original task and are used to extract general features from the input data.
2. **Modifiable Layers**: These are the layers of the model that are adjusted or re-trained during fine-tuning. Modifiable layers learn task-specific features from the new dataset. By focusing on these layers, the model can adapt to the specific requirements of the new task.

Transfer Learning

❑How does Transfer Learning work?

❑Now, one may ask how to determine which layers we need to freeze, and which layers need to train.

❑The answer is simple, the more you want to inherit features from a pre-trained model, the more you have to freeze layers.

Transfer Learning

- Let's consider all situations where the size and dataset of the target task vary from the base network.
- **The target dataset is small and similar to the base network dataset:** Since the target dataset is small, that means we can fine-tune the pre-trained network with the target dataset. But this may lead to a problem of overfitting. Also, there may be some changes in the number of classes in the target task. So, in such a case we remove the fully connected layers from the end, maybe one or two, and add a new fully connected layer satisfying the number of new classes. Now, we freeze the rest of the model and only train newly added layers.
- **The target dataset is large and similar to the base training dataset:** In such cases when the dataset is large, and it can hold a pre-trained model there will be no chance of overfitting. Here, also the last full-connected layer is removed, and a new fully-connected layer is added with the proper number of classes. Now, the entire model is trained on a new dataset. This makes sure to tune the model on a new large dataset keeping the model architecture the same.

Transfer Learning

- Let's consider all situations where the size and dataset of the target task vary from the base network.
- **The target dataset is small and different from the base network dataset:** Since the target dataset is different, using high-level features of the pre-trained model will not be useful. In such a case, remove most of the layers from the end in a pre-trained model, and add new layers a satisfying number of classes in a new dataset. This way we can use low-level features from the pre-trained model and train the rest of the layers to fit a new dataset. Sometimes, it is beneficial to train the entire network after adding a new layer at the end.
- **The target dataset is large and different from the base network dataset:** Since the target network is large and different, the best way is to remove the last layers from the pre-trained network and add layers with a satisfying number of classes, then train the entire network without freezing any layer.

Transfer Learning V1

- In transfer learning, the knowledge of an already trained machine learning model is applied to a different but related problem.
- For example, if you trained a simple classifier to predict whether an image contains a backpack, you could use the knowledge that the model gained during its training to recognize other objects like sunglasses.
- With transfer learning, we basically try to exploit what has been learned in one task to improve generalization in another.
- We transfer the weights that a network has learned at “task A” to a new “task B.”

Transfer Learning V1

- The general idea is to use the knowledge a model has learned from a task with a lot of available labeled training data in a new task that doesn't have much data.
- Instead of starting the learning process from scratch, we start with patterns learned from solving a related task.
- Transfer learning is mostly used in computer vision and natural language processing tasks like sentiment analysis due to the huge amount of computational power required.
- Transfer learning isn't really a machine learning technique, but can be seen as a "design methodology" within the field.
- It is also not exclusive to machine learning. Nevertheless, it has become quite popular in combination with neural networks that require huge amounts of data and computational power.

Transfer Learning V1

❑How Transfer Learning Works

❑In computer vision, for example, neural networks usually try to detect edges in the earlier layers, shapes in the middle layer and some task-specific features in the later layers.

❑In transfer learning, the early and middle layers are used and we only retrain the latter layers.

❑It helps leverage the labeled data of the task it was initially trained on.

❑This process of retraining models is known as fine-tuning.

❑In the case of transfer learning, though, we need to isolate specific layers for retraining.

❑There are then two types of layers to keep in mind when applying transfer learning:

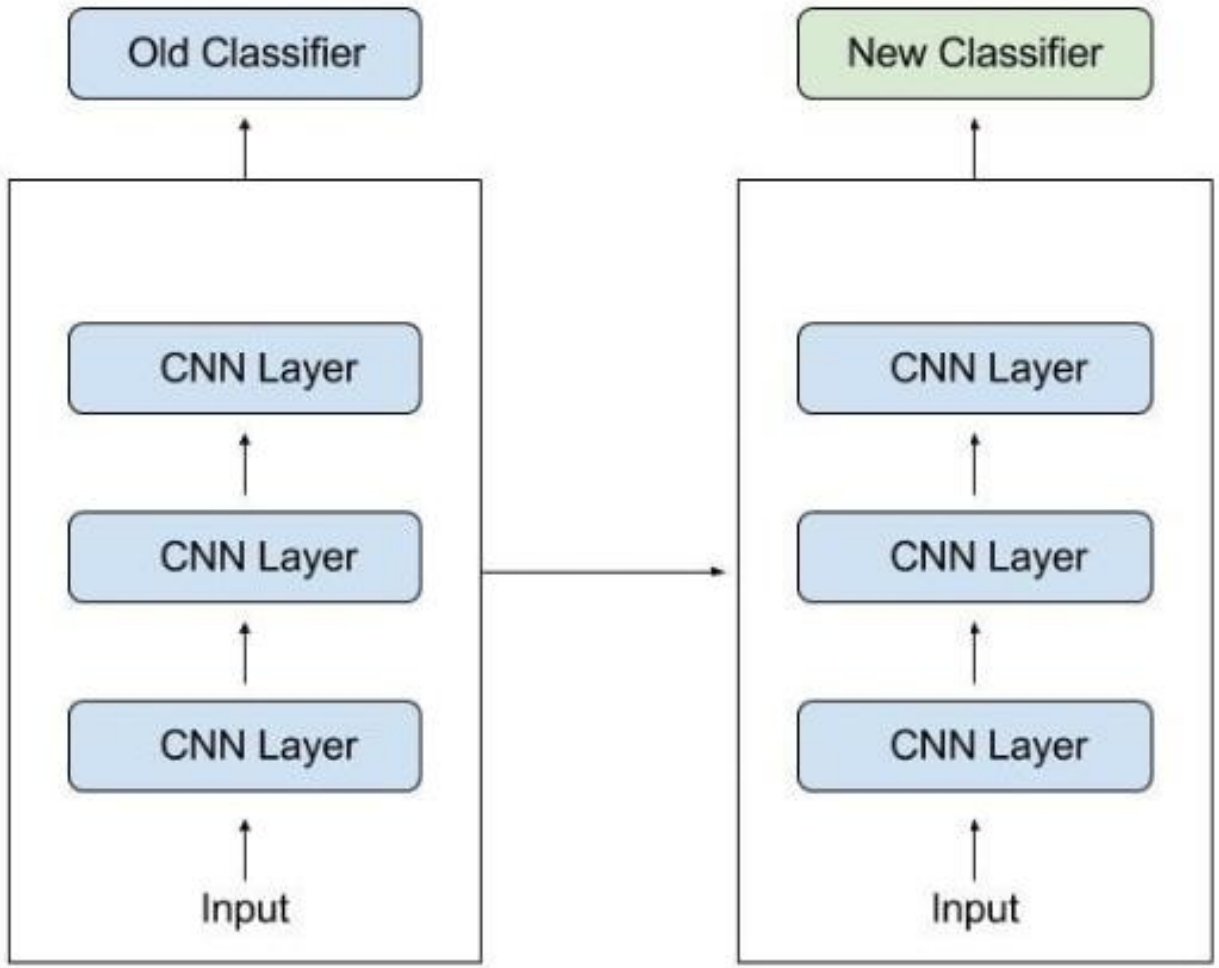
1. **Frozen layers:** Layers that are left alone during retraining and keep their knowledge from a previous task for the model to build on.
2. **Modifiable layers:** Layers that are retrained during fine-tuning, so a model can adjust its knowledge to a new, related task.

Transfer Learning V1

❑How Transfer Learning Works

❑Let's go back to the example of a model trained for recognizing a backpack in an image, which will be used to identify sunglasses.

❑In the earlier layers, the model has learned to recognize objects, so we will only retrain the latter layers to help it learn what separates sunglasses from other objects.



Transfer Learning V1

❑How Transfer Learning Works

❑In transfer learning, we try to transfer as much knowledge as possible from the previous task the model was trained on to the new task at hand.

❑This knowledge can be in various forms depending on the problem and the data.

❑For example, it could be how models are composed, which allows us to more easily identify novel objects.

Transfer Learning V1

❑Why Use Transfer Learning

❑The main advantages of transfer learning are saving training time, improving the performance of neural networks (in most cases) and not needing a lot of data.

❑Usually, a lot of data is needed to train a neural network from scratch, but access to that data isn't always available.

❑With transfer learning, a solid machine learning model can be built with comparatively little training data because the model is already pre-trained.

❑This is especially valuable in natural language processing because mostly expert knowledge is required to create large labeled data sets.

❑Additionally, training time is reduced because it can sometimes take days or even weeks to train a deep neural network from scratch on a complex task.

❑When to Use Transfer Learning

❑As is always the case in machine learning, it is hard to form rules that are generally applicable, but here are some guidelines on when transfer learning might be used:

1. **Lack of training data:** There isn't enough labeled training data to train your network from scratch.
2. **Existing network:** There already exists a network that is pre-trained on a similar task, which is usually trained on massive amounts of data.
3. **Same input:** When task 1 and task 2 have the same input.

❑Approaches to Transfer Learning

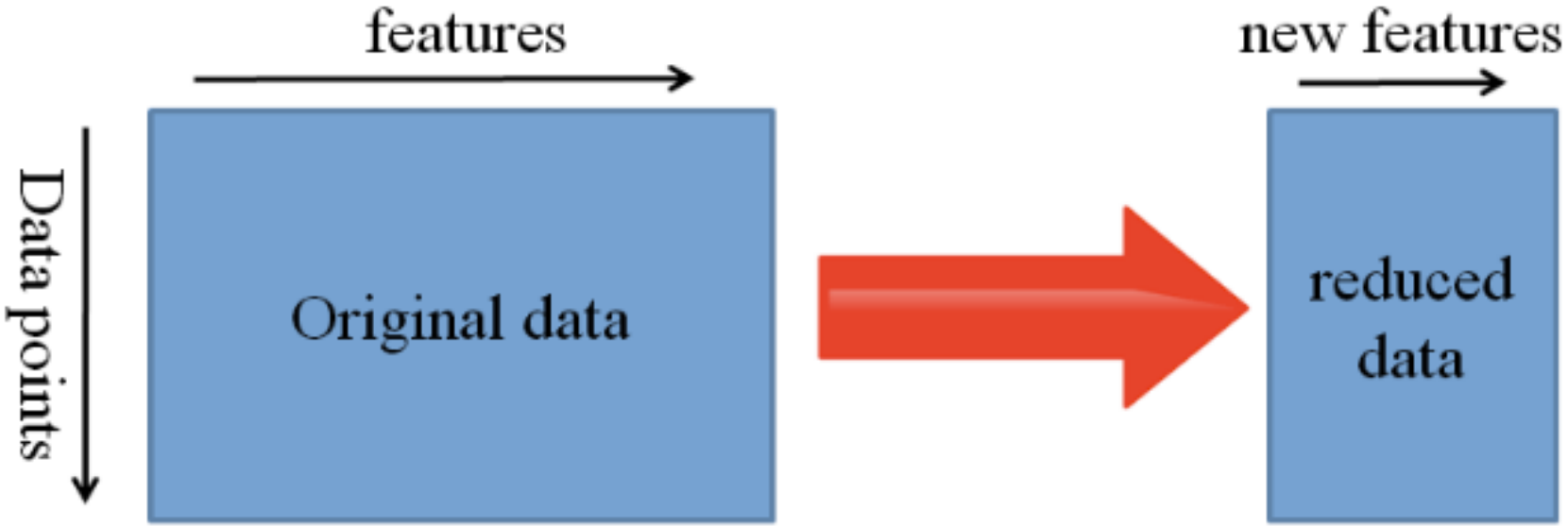
1. Training a Model to Reuse it: Imagine you want to solve task A but don't have enough data to train a deep neural network. One way around this is to find a related task B with an abundance of data. Train the deep neural network on task B and use the model as a starting point for solving task A. Whether you'll need to use the whole model or only a few layers depends heavily on the problem you're trying to solve. If you have the same input in both tasks, possibly reusing the model and making predictions for your new input is an option. Alternatively, changing and retraining different task-specific layers and the output layer is a method to explore.

❑Approaches to Transfer Learning

2. Using a Pre-Trained Model: The second approach is to use an already pre-trained model. There are a lot of these models out there, so make sure to do a little research. How many layers to reuse and how many to retrain depends on the problem. Keras, for example, provides numerous pre-trained models that can be used for transfer learning, prediction, feature extraction and fine-tuning. You can find these models, and also some brief tutorials on how to use them, here. There are also many research institutions that release trained models.This type of transfer learning is most commonly used throughout deep learning.

❑Approaches to Transfer Learning

3. Feature Extraction: Another approach is to use deep learning to discover the best representation of your problem, which means finding the most important features. This approach is also known as representation learning, and can often result in a much better performance than can be obtained with hand-designed representation.



Transfer Learning V2

❑What is Transfer Learning?

❑Transfer learning is a method for feature representation from a pre-trained model facilitating us that we don’t need to train a new model from scratch.

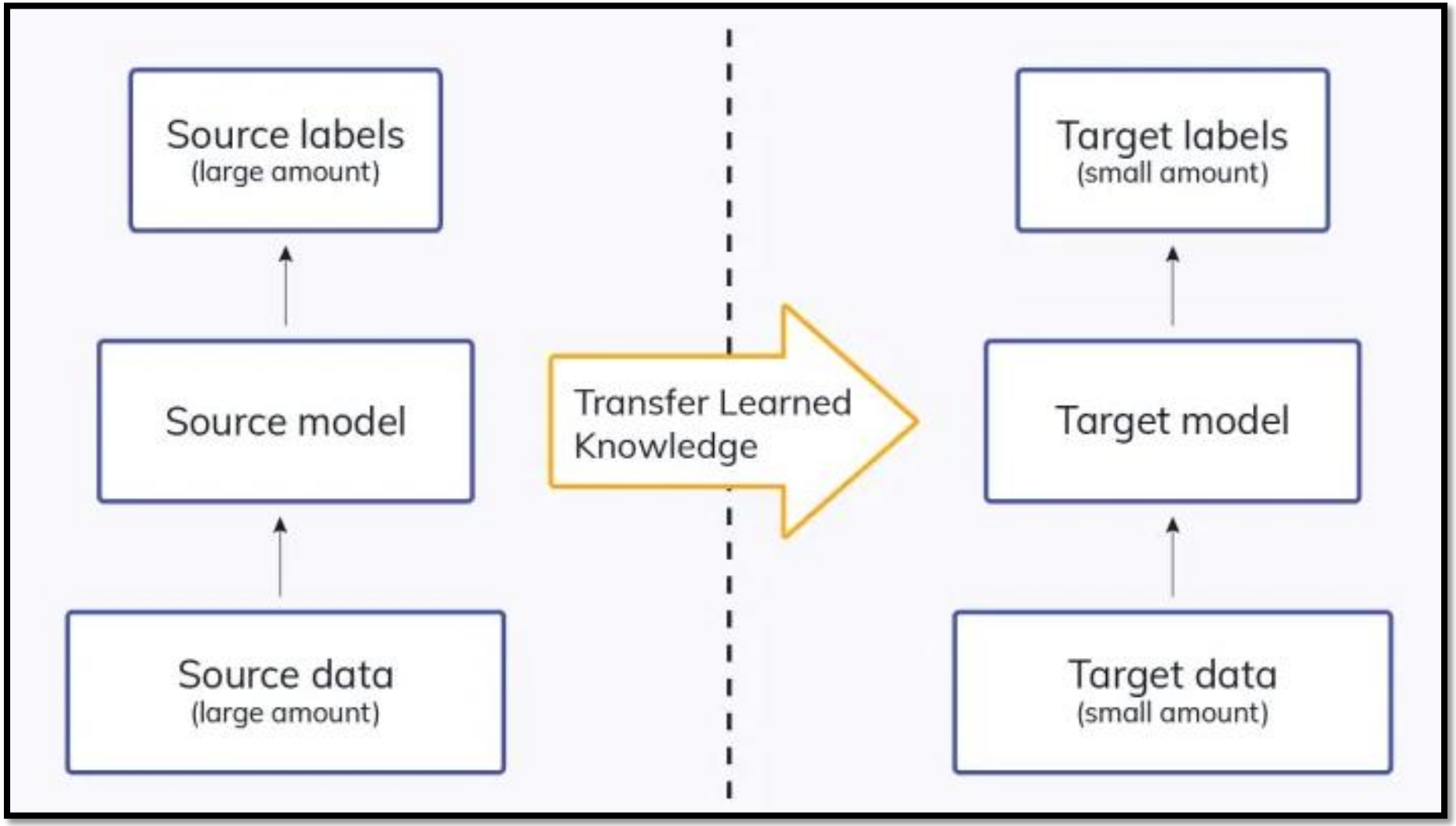
❑A pre-trained model is usually trained on a huge dataset such as ImageNet and the weights obtained from the trained model can be used for any other related application with your custom neural network.

❑These newly built models can directly be used for predictions on relatively new tasks or can be used in training processes for related applications.

❑This approach not only reduces the training time but also lowers the generalization error.

Transfer Learning V2

❑What is Transfer Learning?



Transfer Learning Models

❑Transfer Learning Strategies (Types)

❑The strategy you use to facilitate TL will depend on the domain of the model you are building, the task it needs to complete, and the availability of training data.

Transfer Learning Models

❑Transfer Learning Strategies (Types)

1. Transductive transfer learning:

❑Transductive transfer learning involves transferring knowledge from a specific source domain to a different but related target domain, with the primary focus being on the target domain. It is especially useful when there is little or no labeled data from the target domain.

❑Transductive transfer learning asks the model to make predictions on target data by using previously-gained knowledge. As the target data is mathematically similar to the source data, the model finds patterns and performs faster.

❑For example, consider adapting a sentiment analysis model trained on product reviews to analyze movie reviews. The source domain (product reviews) and the target domain (movie reviews) differ in context and specifics but share similarities in structure and language use. The model quickly learns to apply its understanding of sentiment from the product domain to the movie domain.

❑Transfer Learning Strategies (Types)

2. Inductive transfer learning:

❑Inductive transfer learning is where the source and target domains are the same, but the tasks the model must complete differ. The pre-trained model is already familiar with the source data and trains faster for new functions.

❑An example of inductive transfer learning is in natural language processing (NLP). Models are pre-trained on a large set of texts and then fine-tuned using inductive transfer learning to specific functions like sentiment analysis. Similarly, computer vision models like VGG are pre-trained on large image datasets and then fine-tuned to develop object detection.

❑Transfer Learning Strategies (Types)

3. Unsupervised transfer learning

❑Unsupervised transfer learning uses a strategy similar to inductive transfer learning to develop new abilities. However, you use this form of transfer learning when you only have unlabeled data in both the source and target domains.

❑The model learns the common features of unlabeled data to generalize more accurately when asked to perform a target task. This method is helpful if it is challenging or expensive to obtain labeled source data.

❑For example, consider the task of identifying different types of motorcycles in traffic images. Initially, the model is trained on a large set of unlabeled vehicle images. In this instance, the model independently determines the similarities and distinguishing features among different types of vehicles like cars, buses, and motorcycles. Next, the model is introduced to a small, specific set of motorcycle images. The model performance improves significantly compared to before.

❑Transfer Learning Strategies (Types)

3. Unsupervised transfer learning

❑Unsupervised transfer learning uses a strategy similar to inductive transfer learning to develop new abilities. However, you use this form of transfer learning when you only have unlabeled data in both the source and target domains.

❑The model learns the common features of unlabeled data to generalize more accurately when asked to perform a target task. This method is helpful if it is challenging or expensive to obtain labeled source data.

❑For example, consider the task of identifying different types of motorcycles in traffic images. Initially, the model is trained on a large set of unlabeled vehicle images. In this instance, the model independently determines the similarities and distinguishing features among different types of vehicles like cars, buses, and motorcycles. Next, the model is introduced to a small, specific set of motorcycle images. The model performance improves significantly compared to before.

Transfer Learning Models

❑Transfer Learning Algorithms (Types)

❑Transfer learning involves specialized algorithms to facilitate knowledge transfer. Notable algorithms include:

1. **Feature-based Transfer Learning:** Transfers shared features between domains to enhance target task performance.
2. **Instance-based Transfer Learning:** Adapts source instances to the target domain, useful for dissimilar domains.
3. **Model-based Transfer Learning:** Transfers pre-trained models to adapt to target tasks.
4. **Self-taught Learning:** Trains on a large source domain without target labels to extract useful features.
5. **Domain-adversarial Training:** Learns domain-invariant features by minimizing domain differences.

Transfer Learning Models

❑Transfer Learning Algorithms (Types)

❑Transfer learning involves specialized algorithms to facilitate knowledge transfer. Notable algorithms include:

6. **Zero-shot Learning:** Predicts objects or categories not seen during training using semantic descriptions.
7. **Multi-task Learning:** Trains on multiple tasks simultaneously to leverage shared knowledge.
8. **Inductive Transfer Learning via Matrix Completion:** Adapts knowledge between domains with common objects and different relationships.
9. **Transfer Component Analysis:** Separates shared and domain-specific structures to align domains effectively.

Transfer Learning Models

❑Transfer Learning Algorithms (Types)

❑Transfer learning involves specialized algorithms to facilitate knowledge transfer. Notable algorithms include:

6. **Zero-shot Learning:** Predicts objects or categories not seen during training using semantic descriptions.
7. **Multi-task Learning:** Trains on multiple tasks simultaneously to leverage shared knowledge.
8. **Inductive Transfer Learning via Matrix Completion:** Adapts knowledge between domains with common objects and different relationships.
9. **Transfer Component Analysis:** Separates shared and domain-specific structures to align domains effectively.

❑Transfer Learning Models

❑In the transfer learning, pre-trained models play a central role.

❑These models come with pre-acquired knowledge from various domains and tasks, serving as a starting point for efficient knowledge transfer.

❑Some popular transfer learning models include:

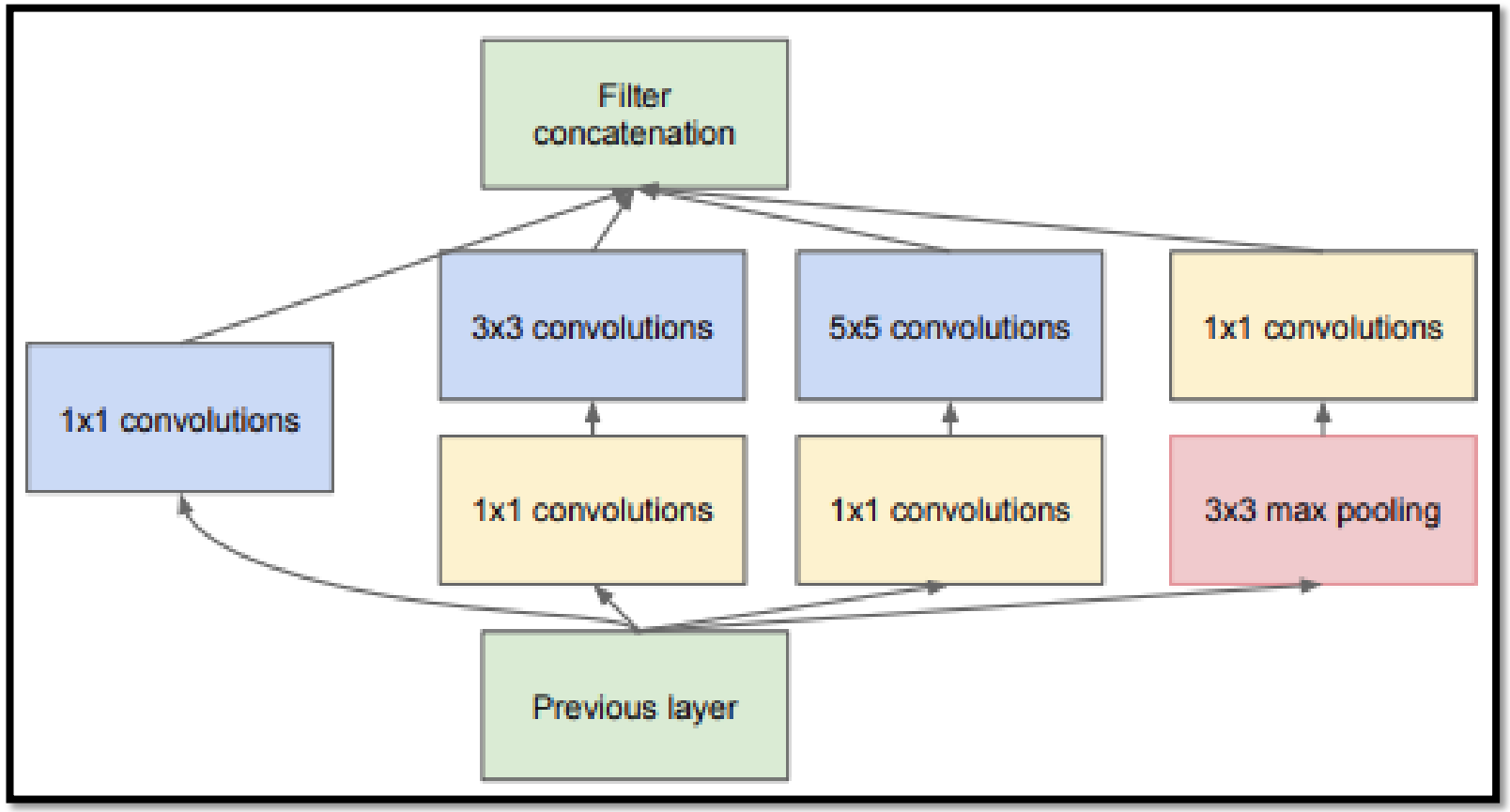
1. **ImageNet Models:** Perfect for computer vision tasks, models like VGG, ResNet, and Inception Excel in image-related applications.
2. **BERT (Bidirectional Encoder Representations from Transformers):** A game-changer in NLP, BERT handles tasks like sentiment analysis and text summarization.
3. **GPT (Generative Pre-trained Transformer):** GPT models are NLP powerhouses, known for natural language generation and understanding.

❑Transfer Learning Models

5. **MobileNet:** Designed for mobile and embedded vision, MobileNet efficiently handles object detection and image classification.
6. **YOLO (You Only Look Once):** Real-time object detection is YOLO's strength, making it valuable for custom solutions.
7. **VGG (Visual Geometry Group Network):** A simple yet effective choice for image classification.
8. **ResNet (Residual Network):** ResNet's deep architecture excels in image classification and object detection.
9. **Inception (GoogLeNet):** Known for resource-efficient computations in computer vision.
10. **Xception:** A model with exceptional performance in image classification.

❑Transfer Learning Models

❑**Inception:** The Inception microarchitecture was introduced by Szegedy in 2014 in their paper Going deeper with convolution the complete architecture with dimension reduction looks like as given below:



❑Transfer Learning Models

❑**Inception:** The goal of this module is to act as a multi-level *feature extractor* by computing 1×1, 3×3, and 5×5 convolution within the same module of the network.

❑The output of these filters is then stacked along the channel dimension and before being fed into the next layer in the network.

❑The architecture of this model includes:

1. 1×1 convolution with 128 filters for dimensions and reductions and rectified linear activations
2. Fully connected layer with 1024 units and a rectified linear activation
3. Dropout layer with 70% ratio
4. Linear layer with softmax loss as the classifier
5. Originally this architecture was called GoogleNet subsequently it has simply been called InceptionN where N refers to the version of the model.

❑Transfer Learning Models

❑Xception

❑This model was proposed by Francois *Chollet* the creator and maintainer of the Keras library.

❑The Xception is an extension of inception architecture that replaces the standard inception model with depth wise separable convolutions.

❑From the below architecture, it is clear that Xception is a linear stack of depthwise separable convolution layers with residual connections.

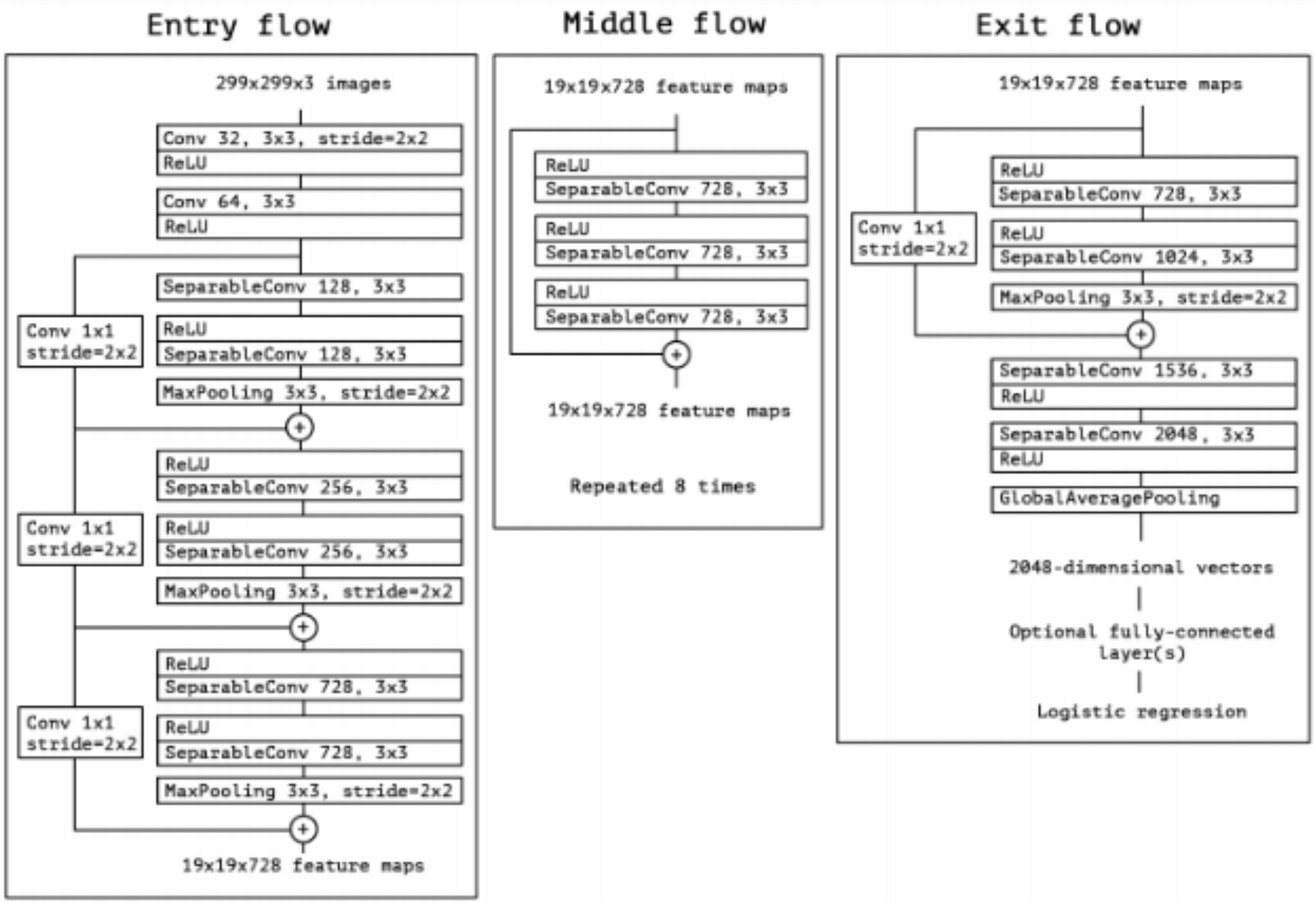
❑This makes architecture very easy to define and modify; it takes only 40 lines of code by using high-level APIs such as Keras or Tensorflow.

❑As you can see below, the data first goes through the Entry flow, then through the middle flow which is repeated eight times and finally through the exit flow.

❑All convolutional layers follow batch normalization.

Transfer Learning Models

- Transfer Learning Models
- Xception



Transfer Learning Models

❑Transfer Learning Models

❑VGG Family

❑This model was first proposed by Zisserman and Simonyan from the Visual Geometry Group (VGG) of the University of Oxford in their paper *Very deep convolutional networks for large scale image recognition.*

❑The network is recognized by its simplicity using only a stack of 3×3 convolutional layers on top of each increasing depth and volume size handled by the max-pooling layers.

❑Two fully connected layers each with 4096 nodes are then followed by a softmax layer.

❑During training the input to the model is fixed-sized RGB images and only preprocessing is done at the training is subtracting the mean RGB values computed on the training set for each pixel.

❑The image is then passed through the stack of convolutional layers where it uses filter very small receptive files of 3×3 which is fair enough to capture the smallest notation.

❑Spatial pooling is carried out by five max-pooling layers which follow some convolutional layer over a 2×2 pixel with strides of 2 and the last is usually the same for all architectures, i.e., softmax.

❑Transfer Learning Models

❑VGG Family

❑As the depth of the network increases, it becomes slow at training and network architectures themselves also become quite large.

❑The architectures of all the VGG variants are shown below with their parameters:

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

❑What are transfer learning strategies in generative AI?

❑Transfer learning strategies are critical for generative AI adoption in various industries.

❑Organizations can customize existing foundation models without having to train new ones on billions of data parameters at scale.

❑The following are some transfer learning strategies used in generative AI.

1. Domain adversarial training

❑Domain adversarial training involves training a foundation model to produce data that is indistinguishable from real data in the target domain. This technique typically employs a discriminator network, as seen in generative adversarial networks, that attempts to distinguish between true data and generated data. The generator learns to create increasingly realistic data.

❑For example, in image generation, a model trained on photographs might be adapted to generate artwork. The discriminator helps ensure the generated artwork is stylistically consistent with the target domain.

❑What are transfer learning strategies in generative AI?

2. Teacher-student learning

❑Teacher-student learning involves a larger and more complex “teacher” model teaching a smaller and simpler “student” model. The student model learns to mimic the teacher model's behavior, effectively transferring knowledge. This is useful for deploying large generative models in resource-constrained environments.

❑For example, a large language model (LLM) could serve as a teacher to a smaller model, transferring its language generation capabilities. This would allow the smaller model to generate high-quality text with less computational overhead.

3. Feature disentanglement

❑Feature disentanglement in generative models involves separating different aspects of data, such as content and style, into distinct representations. This enables the model to manipulate these aspects independently in the transfer learning process.

❑For example, in a face generation task, a model might learn to disentangle facial features from artistic style. This would allow it to generate portraits in various artistic styles while maintaining the subject's likeness.

❑What are transfer learning strategies in generative AI?

4. Cross-modal transfer learning

❑Cross-modal transfer learning involves transferring knowledge between different modalities, like text and images. Generative models can learn representations applicable across these modalities. A model trained on textual descriptions and corresponding images might learn to generate relevant images from new text descriptions, effectively transferring its understanding from text to image.

5. Zero-shot and few-shot learning

❑In zero-shot and few-shot learning, generative models are trained to perform tasks or generate data for which they have seen few or no examples of during training. This is achieved by learning rich representations that generalize well. For example, a generative model might be trained to create images of animals. Using few-shot learning, it could generate images of a rarely seen animal by understanding and combining features from other animals.

❑What are transfer learning strategies in generative AI?

4. Cross-modal transfer learning

❑Cross-modal transfer learning involves transferring knowledge between different modalities, like text and images. Generative models can learn representations applicable across these modalities. A model trained on textual descriptions and corresponding images might learn to generate relevant images from new text descriptions, effectively transferring its understanding from text to image.

5. Zero-shot and few-shot learning

❑In zero-shot and few-shot learning, generative models are trained to perform tasks or generate data for which they have seen few or no examples of during training. This is achieved by learning rich representations that generalize well. For example, a generative model might be trained to create images of animals. Using few-shot learning, it could generate images of a rarely seen animal by understanding and combining features from other animals.

Generative Adversarial Network and Their Variants

❑What is a GAN?

❑A generative adversarial network (GAN) is a deep learning architecture.

❑It trains two neural networks to compete against each other to generate more authentic new data from a given training dataset.

❑For instance, you can generate new images from an existing image database or original music from a database of songs.

❑A GAN is called adversarial because it trains two different networks and pits them against each other.

❑One network generates new data by taking an input data sample and modifying it as much as possible.

❑The other network tries to predict whether the generated data output belongs in the original dataset.

❑In other words, the predicting network determines whether the generated data is fake or real.

❑The system generates newer, improved versions of fake data values until the predicting network can no longer distinguish fake from original.

❑What are some use cases of generative adversarial networks?

❑The GAN architecture has several applications across different industries.

❑Next, we give some examples.

1. Generate images

▪Generative adversarial networks create realistic images through text-based prompts or by modifying existing images.

▪They can help create realistic and immersive visual experiences in video games and digital entertainment.

▪GAN can also edit images—like converting a low-resolution image to a high resolution or turning a black-and-white image to color.

▪It can also create realistic faces, characters, and animals for animation and video.

❑What are some use cases of generative adversarial networks?

2. Generate training data for other models

- In machine learning (ML), data augmentation artificially increases the training set by creating modified copies of a dataset using existing data.
- You can use generative models for data augmentation to create synthetic data with all the attributes of real-world data.
- For instance, it can generate fraudulent transaction data that you then use to train another fraud-detection ML system.
- This data can teach the system to accurately distinguish between suspicious and genuine transactions.

❑What are some use cases of generative adversarial networks?

3. Complete missing information

❑Sometimes, you may want the generative model to accurately guess and complete some missing information in a dataset.

❑For instance, you can train GAN to generate images of the surface below ground (sub-surface) by understanding the correlation between surface data and underground structures.

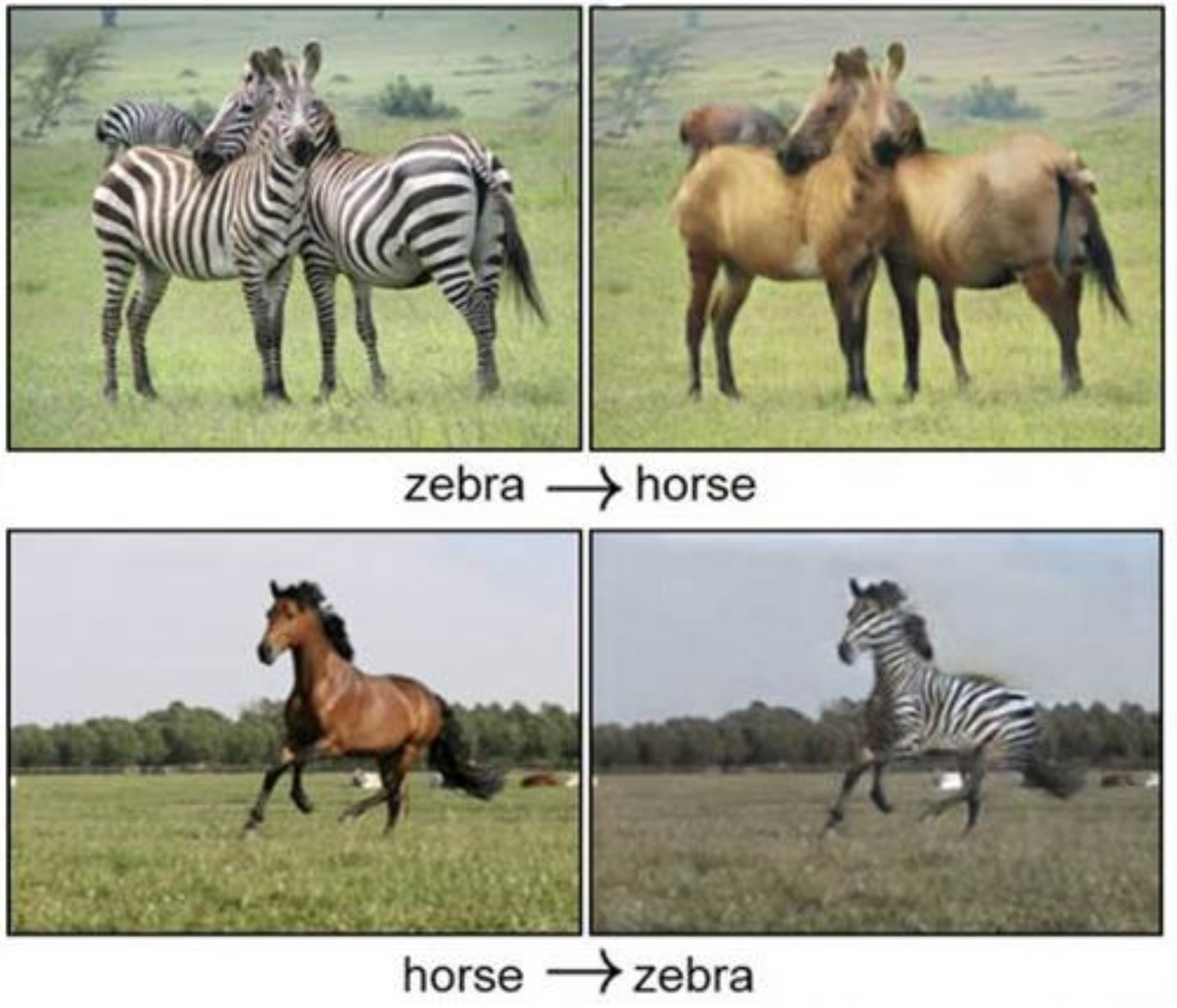
❑By studying known sub-surface images, it can create new ones using terrain maps for energy applications like geothermal mapping or carbon capture and storage.

4. Generate 3D models from 2D data

❑GAN can generate 3D models from 2D photos or scanned images.

❑For instance, in healthcare, GAN combines X-rays and other body scans to create realistic images of organs for surgical planning and simulation.

❑What are some use cases of generative adversarial networks?



Generative Adversarial Network and Their Variants

❑How does a generative adversarial network work?

❑A generative adversarial network system comprises two deep neural networks—the generator network and the discriminator network.

❑Both networks train in an adversarial game, where one tries to generate new data and the other attempts to predict if the output is fake or real data.

❑How does a generative adversarial network work?

❑GAN works as follows.

1. The generator neural network analyzes the training set and identifies data attributes
2. The discriminator neural network also analyzes the initial training data and distinguishes between the attributes independently
3. The generator modifies some data attributes by adding noise (or random changes) to certain attributes
4. The generator passes the modified data to the discriminator
5. The discriminator calculates the probability that the generated output belongs to the original dataset
6. The discriminator gives some guidance to the generator to reduce the noise vector randomization in the next cycle

Generative Adversarial Network and Their Variants

❑How does a generative adversarial network work?

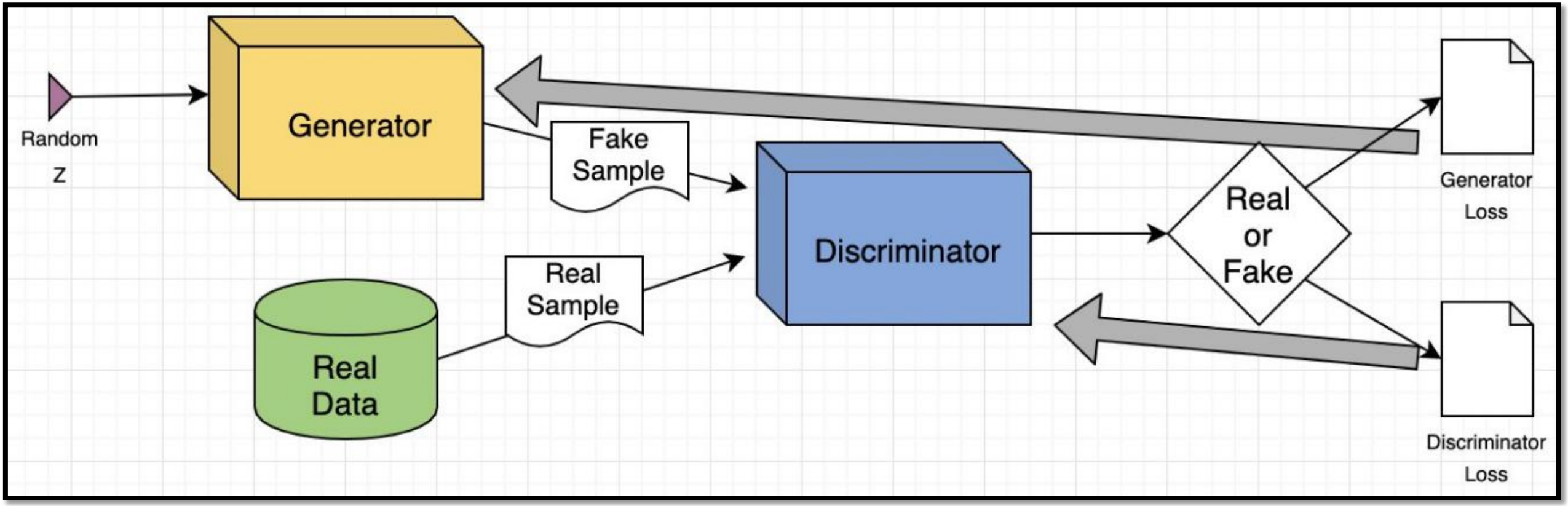
❑The generator attempts to maximize the probability of mistake by the discriminator, but the discriminator attempts to minimize the probability of error.

❑In training iterations, both the generator and discriminator evolve and confront each other continuously until they reach an equilibrium state.

❑In the equilibrium state, the discriminator can no longer recognize synthesized data. At this point, the training process is over.

Generative Adversarial Network and Their Variants

❑How does a generative adversarial network work?



❑GAN training example

- ❑Let's contextualize the above with an example of the GAN model in image-to-image translation.
- ❑Consider that the input image is a human face that the GAN attempts to modify.
- ❑For example, the attributes can be the shapes of eyes or ears.
- ❑Let's say the generator changes the real images by adding sunglasses to them. The discriminator receives a set of images, some of real people with sunglasses and some generated images that were modified to include sunglasses.
- ❑If the discriminator can differentiate between fake and real, the generator updates its parameters to generate even better fake images.
- ❑If the generator produces images that fool the discriminator, the discriminator updates its parameters.
- ❑Competition improves both networks until equilibrium is reached.

Generative Adversarial Network and Their Variants

❑What are the types of generative adversarial networks?

❑There are different types of GAN models depending on the mathematical formulas used and the different ways the generator and discriminator interact with each other.

❑We give some commonly used models next, but the list is not comprehensive.

❑There are numerous other GAN types—like StyleGAN, CycleGAN, and DiscoGAN—that solve different types of problems.

1. **Vanilla GAN:** This is the basic GAN model that generates data variation with little or no feedback from the discriminator network. A vanilla GAN typically requires enhancements for most real-world use cases.
2. **Conditional GAN:** A conditional GAN (cGAN) introduces the concept of conditionality, allowing for targeted data generation. The generator and discriminator receive additional information, typically as class labels or some other form of conditioning data.

❑What are the types of generative adversarial networks?

3. **Deep Convolutional GAN:** Recognizing the power of convolutional neural networks (CNNs) in image processing, Deep convolutional GAN (DCGAN) integrates CNN architectures into GANs. With DCGAN, the generator uses transposed convolutions to upscale data distribution, and the discriminator also uses convolutional layers to classify data. The DCGAN also introduces architectural guidelines to make training more stable.
4. **Super-resolution GANS (SRGANs):** focus on upscaling low-resolution images to high resolution. The goal is to enhance images to a higher resolution while maintaining image quality and details.
5. **Laplacian Pyramid GANs (LAPGANs):** address the challenge of generating high-resolution images by breaking down the problem into stages. They use a hierarchical approach, with multiple generators and discriminators working at different scales or resolutions of the image. The process begins with generating a low-resolution image that improves in quality over progressive GAN stages.

Region Based Convolutional Neural Network (R-CNN)

- R-CNN (Region-based Convolutional Neural Network) was introduced by Ross Girshick et al. in 2014.
- R-CNN revolutionized object detection by combining the strengths of region proposal algorithms and deep learning, leading to remarkable improvements in detection accuracy and efficiency.
- Traditional Convolutional Neural Networks (CNNs) with fully connected layers struggle to handle the frequency of object occurrences and multi-object scenarios.
- A brute-force approach using a sliding window to select regions and apply CNNs is computationally expensive, especially since objects can vary significantly in size and aspect ratio.

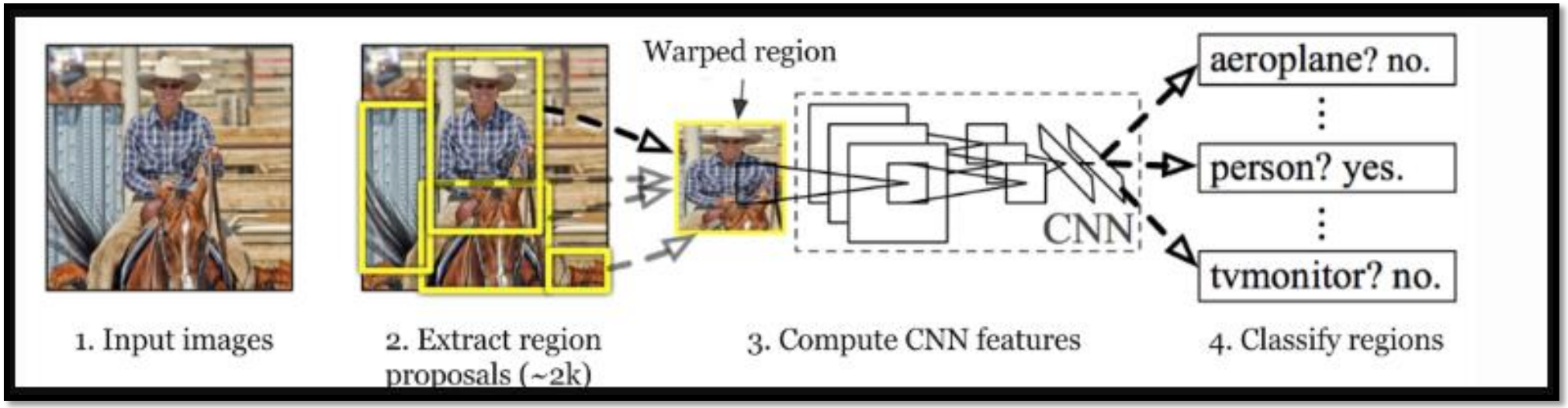
Region Based Convolutional Neural Network (R-CNN)

- To tackle the challenges of object detection, Ross Girshick introduced R-CNN.
- This approach utilizes a selective search algorithm to generate approximately 2,000 region proposals, which are then processed through a Convolutional Neural Network (CNN) to extract features.
- These features are classified using a Support Vector Machine (SVM), while a bounding box regressor is employed to improve localization accuracy.
- R-CNN identifies and localizes objects in images by proposing Regions of Interest (RoIs) and classifying them through the CNN.
- The object detection framework starts with an input image containing potential objects and employs a Region Proposal Network (RPN), like Selective Search, to generate bounding boxes likely to contain objects.

Region Based Convolutional Neural Network (R-CNN)

- Each proposed region is resized and fed into a pre-trained CNN, such as *AlexNet or VGG16*, to extract feature representations.
- These features are then classified by the SVM into predefined categories or designated as background.
- To refine localization further, a bounding box regression model adjusts the coordinates of each box, aligning them more closely with the actual object boundaries.
- This systematic process effectively combines proposal generation, feature extraction, classification, and bounding box refinement, enabling accurate object detection.

Region Based Convolutional Neural Network (R-CNN)



Region Based Convolutional Neural Network (R-CNN)

❑Key Features of R-CNNs

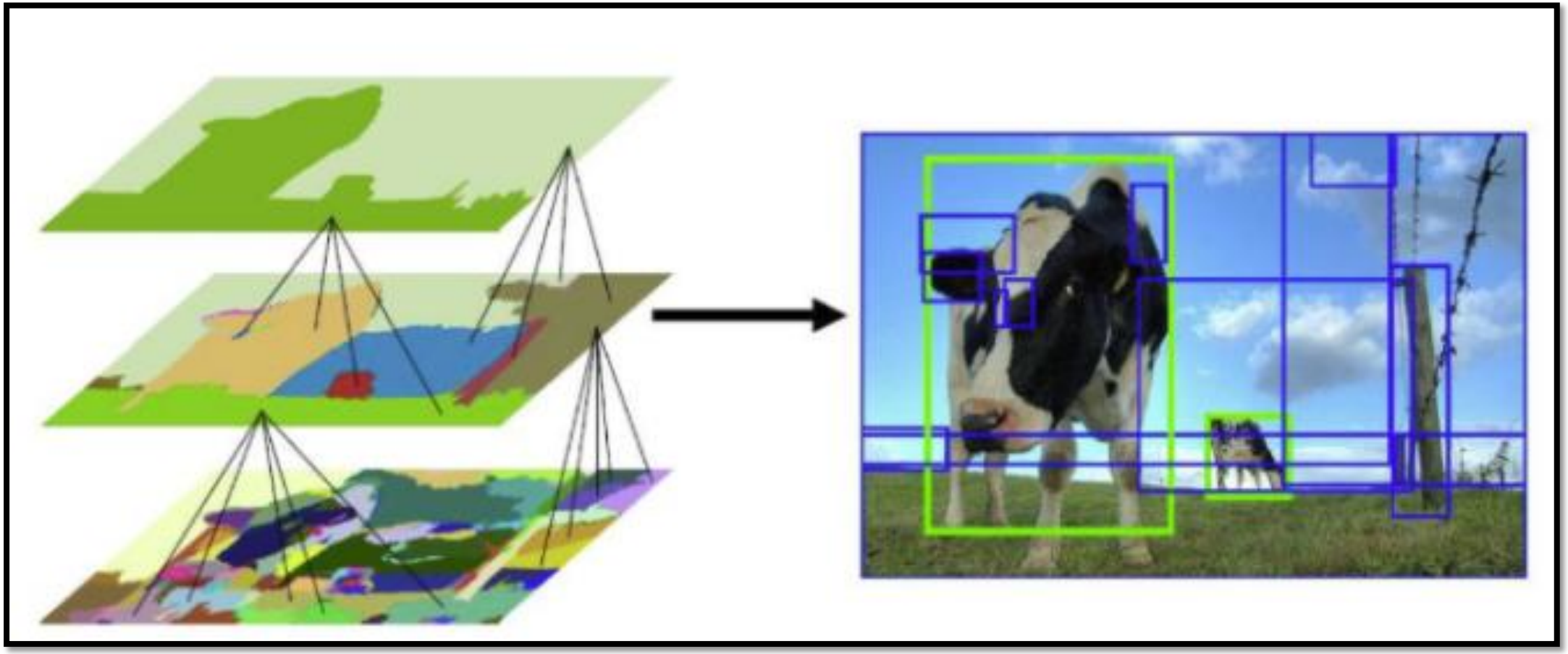
1. Region Proposals:

- R-CNNs begin by generating region proposals, which are smaller sections of the image that may contain the objects we are searching for.
- The algorithm employs a method called selective search, a greedy approach that generates approximately 2,000 region proposals per image.
- Selective search effectively balances the number of proposals while maintaining high object recall, ensuring efficient object detection.
- By limiting the number of regions for detailed analysis, this method enhances the overall performance of the R-CNN in detecting objects within images.

Region Based Convolutional Neural Network (R-CNN)

❑Key Features of R-CNNs

1. Region Proposals:



Region Based Convolutional Neural Network (R-CNN)

❑Key Features of R-CNNs

2. **Selective Search:** is a greedy algorithm that generates region proposals by combining smaller segmented regions. It takes an image as input and produces region proposals that are crucial for object detection. This method offers significant advantages over random proposal generation by limiting the number of proposals to approximately 2,000 while ensuring high object recall.

❑ **Algorithm Steps:**

I. **Generate Initial Segmentation**: The algorithm starts by performing an initial sub-segmentation of the input image.

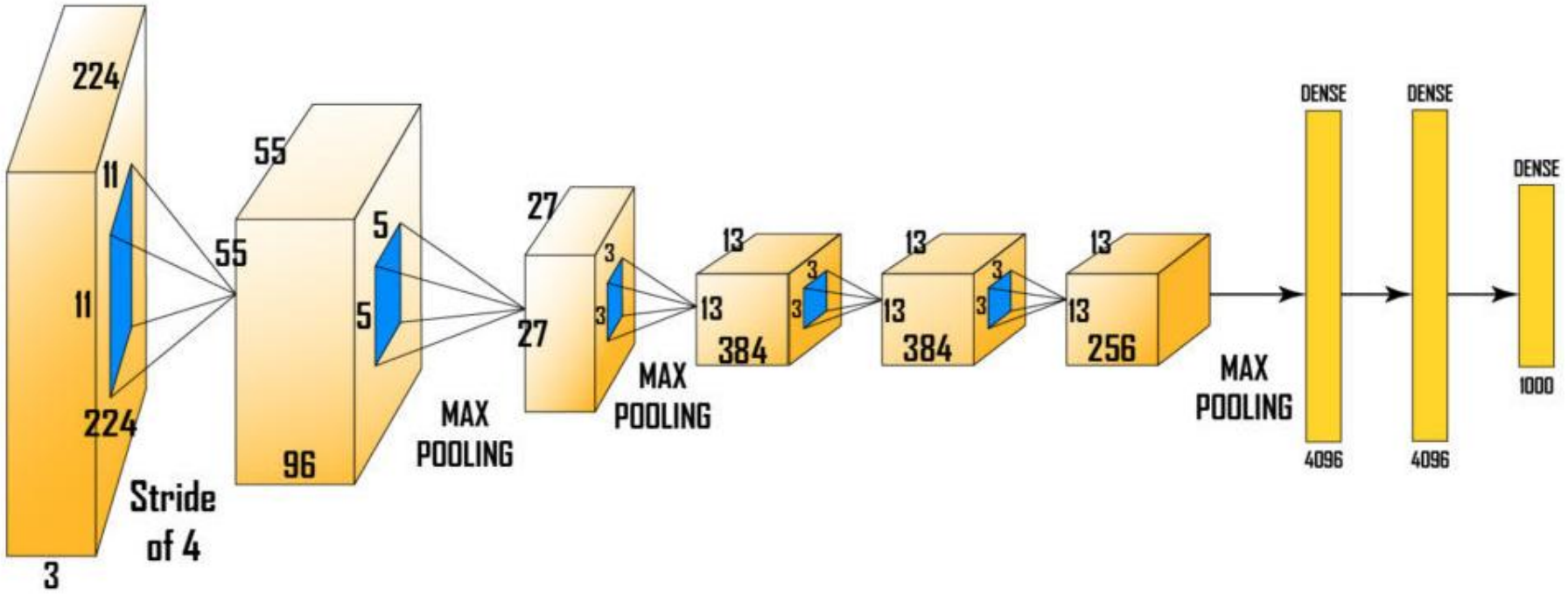
II. **Combine Similar Regions**: It then recursively combines similar bounding boxes into larger ones. Similarities are evaluated based on factors such as color, texture, and region size.

III. **Generate Region Proposals**: Finally, these larger bounding boxes are used to create region proposals for object detection.

Region Based Convolutional Neural Network (R-CNN)

❑Key Features of R-CNNs

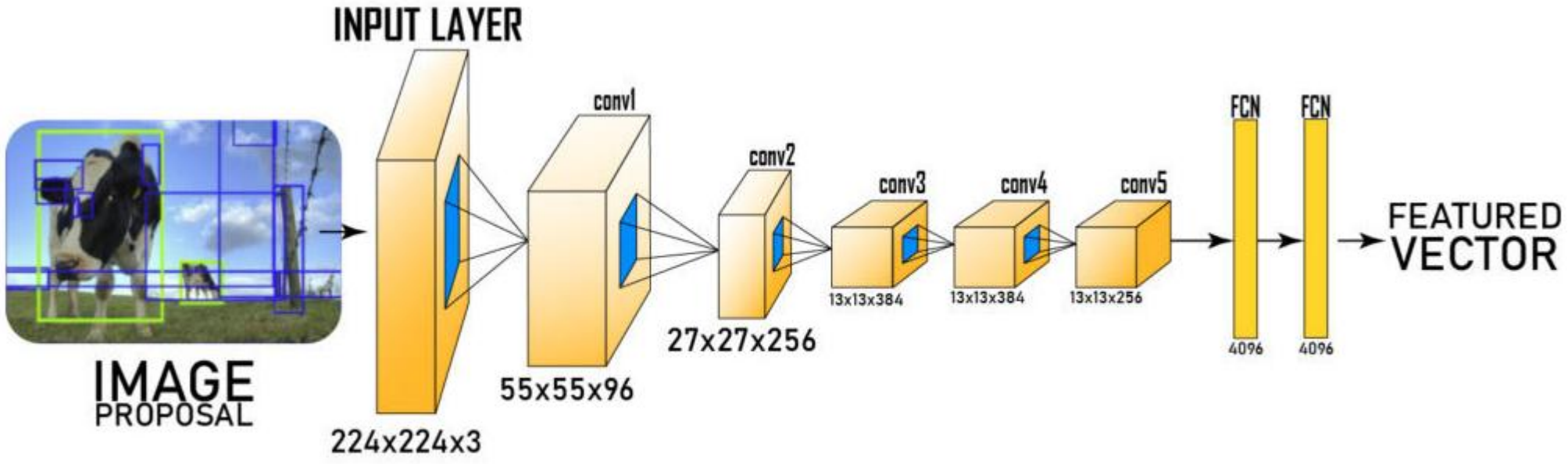
3. **Input Preparation in R-CNN:** After generating the region proposals, these regions are warped into a uniform square shape to match the input dimensions required by the CNN model. In this case, we use the pre-trained AlexNet model, which was considered the state-of-the-art CNN for image classification at the time.



Region Based Convolutional Neural Network (R-CNN)

❑Key Features of R-CNNs

3. **Input Preparation in R-CNN:** The input size for AlexNet is (227, 227, 3), meaning each input image must be resized to these dimensions. Consequently, whether the region proposals are small or large, they need to be adjusted accordingly to fit the specified input size. From the above architecture, we remove the final softmax layer to obtain a (1, 4096) feature vector. This feature vector is then fed into both the Support Vector Machine (SVM) for classification and the bounding box regressor for improved localization.



❑Key Features of R-CNNs

4. SVM (Support Vector Machine):

- The feature vector generated by the CNN is then utilized by a binary Support Vector Machine (SVM), which is trained independently for each class.
- This SVM model takes the feature vector produced by the previous CNN architecture and outputs a confidence score indicating the likelihood of an object being present in that region.
- However, a challenge arises during the training process with the SVM: it requires the AlexNet feature vectors for each class. As a result, we cannot train AlexNet and the SVM independently and in parallel.

Region Based Convolutional Neural Network (R-CNN)

❑Key Features of R-CNNs

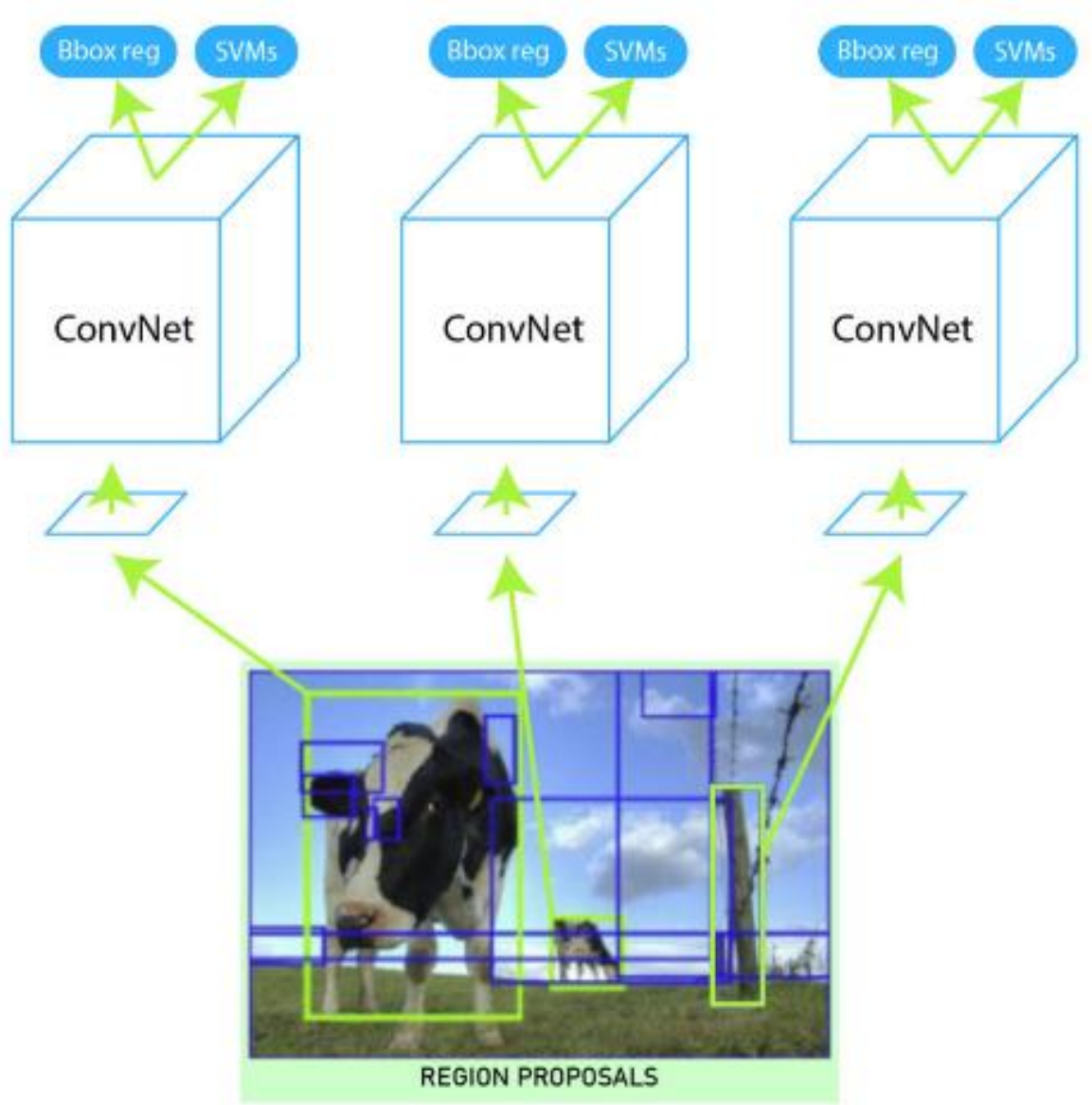
5. **Bounding Box Regressor:** To accurately locate the bounding box within the image, we utilize a scale-invariant linear regression model known as the bounding box regressor.

- For training this model, we use pairs of predicted and ground truth values for four dimensions of localization: (, , ,h)(x,y,w,h).
- Here, x and y represent the pixel coordinates of the center of the bounding box, while w and hh indicate the width and height of the bounding boxes, respectively.
- This method enhances the Mean Average Precision (mAP) of the results by 3-4%.

Region Based Convolutional Neural Network (R-CNN)

❑Key Features of R-CNNs

5. Bounding Box Regressor:



Region Based Convolutional Neural Network (R-CNN)

❑Key Features of R-CNNs

❑To further optimize detection, R-CNNs apply Non-Maximum Suppression (NMS):

❑Remove proposals with confidence scores below a threshold (e.g., 0.5).

❑Select the highest-probability region among candidates for each object.

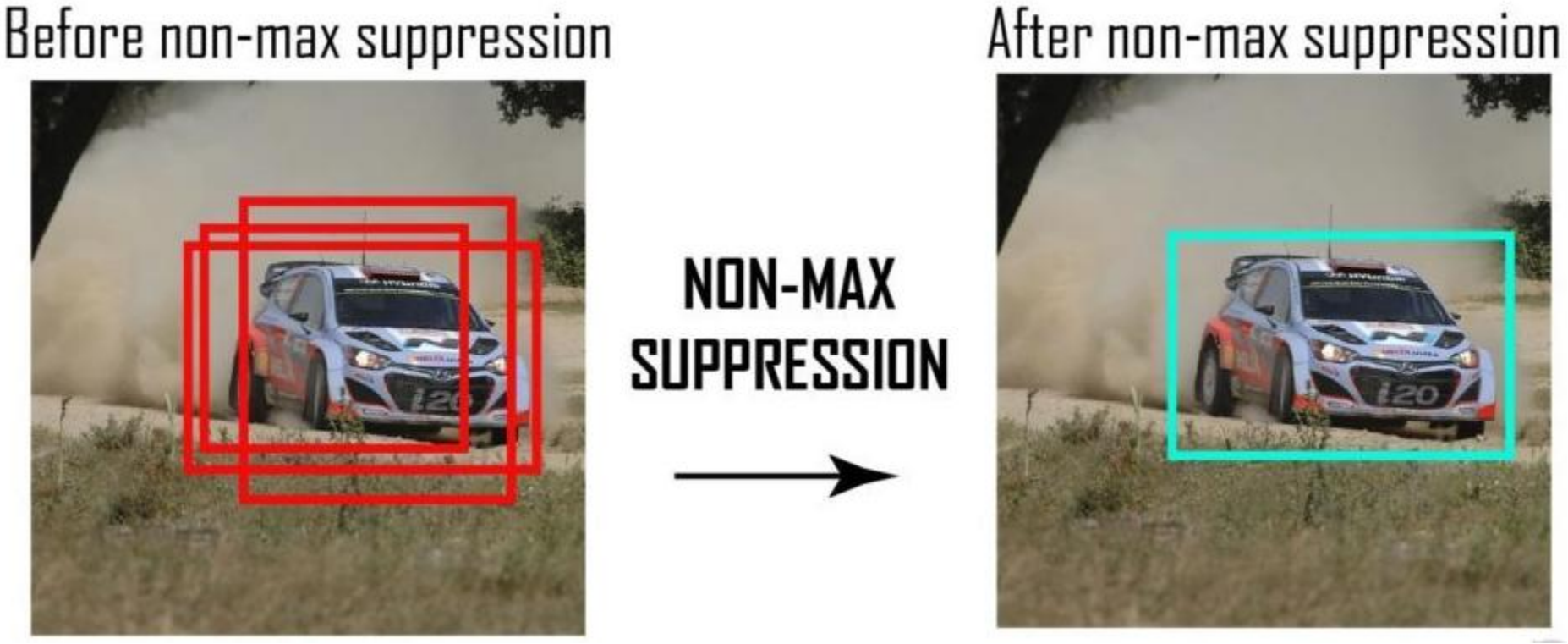
❑Discard overlapping regions with an IoU (Intersection over Union) above 0.5 to eliminate duplicate detections, where IoU is defined as:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Region Based Convolutional Neural Network (R-CNN)

❑Key Features of R-CNNs

❑By combining region proposals, selective search, CNN-based feature extraction, SVM classification, and bounding box refinement, R-CNN achieves high accuracy in object detection, making it suitable for various applications.



Region Based Convolutional Neural Network (R-CNN)

❑**Challenges of R-CNN:** R-CNN faces several challenges in its implementation:

1. **Rigid Selective Search Algorithm**: The selective search algorithm is inflexible and does not involve any learning. This rigidity can result in poor region proposal generation for object detection.
2. **Time-Consuming Training**: With approximately 2,000 candidate proposals, training the network becomes time-intensive. Additionally, multiple components need to be trained separately, including the CNN architecture, SVM model, and bounding box regressor. This multi-step training process slows down implementation.
3. **Inefficiency for Real-Time Applications**: R-CNN is not suitable for real-time applications, as it takes around 50 seconds to process a single image with the bounding box regressor.
4. **Increased Memory Requirements**: Storing feature maps for all region proposals significantly increases the disk memory needed during the training phase.

Region Based Convolutional Neural Network (R-CNN)

❑Evolution of R-CNN: Fast R-CNN and Mask R-CNN

Following the introduction of R-CNN, several variations emerged to address its limitations:

1. Fast R-CNN: Introduced by Ross Girshick in 2015, Fast R-CNN optimizes the R-CNN architecture by sharing computations across proposals. Key improvements include:

- **Single Stage Processing**: Instead of extracting features for each region proposal independently, Fast R-CNN processes the entire image once through the CNN to generate a feature map. The region proposals are then extracted from this shared feature map.
- **Softmax Classifier**: Fast R-CNN replaces the SVM with a softmax classifier, allowing for end-to-end training of the network.
- **Improved Bounding Box Regression**: Fast R-CNN enhances the bounding box regression process, leading to better localization accuracy.

Region Based Convolutional Neural Network (R-CNN)

❑Evolution of R-CNN: Fast R-CNN and Mask R-CNN

2. Faster R-CNN: Faster R-CNN, also introduced in 2015, further advances the R-CNN framework by incorporating a Region Proposal Network (RPN). Key features include:

- **Region Proposal Network**: The RPN generates high-quality region proposals directly from the feature maps produced by the CNN, eliminating the need for selective search.
- **Shared Convolutional Features**: Both the RPN and the detection network share the convolutional features, significantly reducing computation time.
- **Improved Speed**: Faster R-CNN achieves real-time processing speeds of around 0.1 seconds per image while maintaining high detection accuracy.

Region Based Convolutional Neural Network (R-CNN)

❑Evolution of R-CNN: Fast R-CNN and Mask R-CNN

3. Mask R-CNN: Building upon Faster R-CNN, Mask R-CNN was introduced in 2017 to extend the model to perform instance segmentation. Key features include:

- **Segmentation Masks**: In addition to bounding boxes, Mask R-CNN predicts a segmentation mask for each detected object, providing pixel-level accuracy.
- **Feature Pyramid Networks (FPN)**: Mask R-CNN incorporates FPNs to improve performance on objects at different scales, enhancing detection accuracy for small objects.
- **RoIAlign**: This technique replaces RoIPooling to address misalignment issues, ensuring better feature extraction for each region of interest.

Region Based Convolutional Neural Network (R-CNN)

❑Evolution of R-CNN: Fast R-CNN and Mask R-CNN

4. Cascade R-CNN: Introduced in 2018, Cascade R-CNN implements a multi-stage object detection framework to improve detection performance. Key aspects include:

- **Multi-Stage Detection**: Cascade R-CNN employs a series of detectors operating at different stages, progressively refining the proposals and improving localization accuracy.
- **Improved Recall and Precision**: By addressing the trade-off between recall and precision at each stage, the model enhances overall detection performance, especially on challenging datasets.

Region Based Convolutional Neural Network (R-CNN)

❑Applications of R-CNN

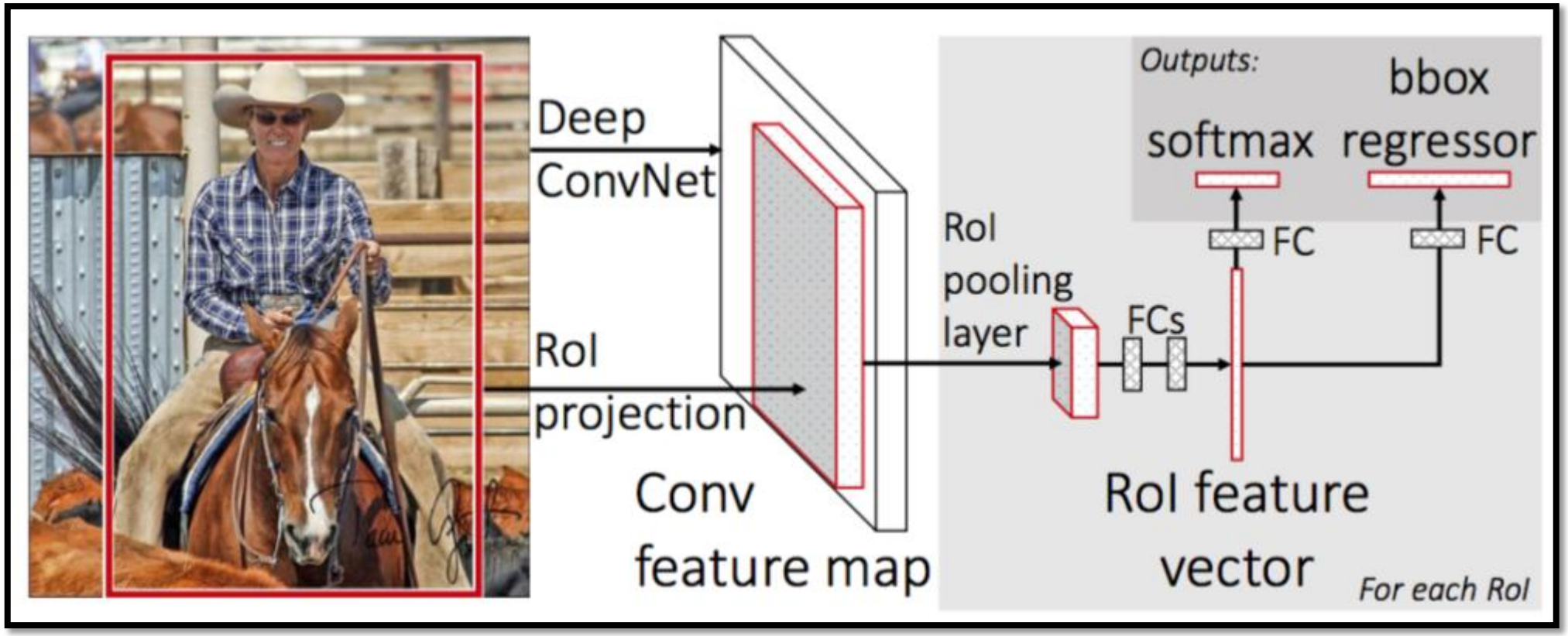
1. **Autonomous Vehicles**: R-CNN can detect and classify various objects on the road, such as pedestrians, other vehicles, and traffic signs, contributing to safer navigation.
2. **Surveillance Systems**: In security applications, R-CNN can identify suspicious activities by detecting and classifying individuals and objects in real-time.
3. **Medical Imaging**: R-CNN is used in medical applications to identify anomalies in medical scans, assisting in early diagnosis and treatment.
4. **Augmented Reality**: R-CNN can enable object recognition in augmented reality applications, enhancing user experiences by overlaying digital information on the real world.

Fast RCNN

- Before discussing Fast R-CNN, let's look at the challenges faced by R-CNN.
- The training of R-CNN is very slow because each part of the model such as (CNN, SVM classifier, and bounding box) requires training separately and cannot be paralleled.
- Also, in R-CNN we need to forward and pass every region proposal through the Deep Convolution architecture (that's up to ~2000 region proposals per image).
- That explains the amount of time taken to train this model
- The testing time of inference is also very high.
- It takes 49 seconds to test an image in R-CNN (along with selective search region proposal generation).

Fast RCNN

- Fast R-CNN works to solve these problems. Let's look at the architecture of Fast R-CNN.



Fast RCNN

- First, we generate the region proposal from a selective search algorithm.
- This selective search algorithm generates up to approximately 2000 region proposals.
- These region proposals (RoI projections) combine with input images passed into a CNN network.
- This CNN network generates the convolution feature map as output.
- Then for each object proposal, a Region of Interest (RoI) pooling layer extracts the feature vector of fixed length for each feature map.
- Every feature vector is then passed into twin layers of softmax classifier and Bbox regression for classification of region proposal and improve the position of the bounding box of that object.

Fast RCNN

❑CNN Network of Fast R-CNN

❑Fast R-CNN is experimented with three pre-trained ImageNet networks each with 5 max-pooling layers and 5-13 convolution layers (such as VGG-16).

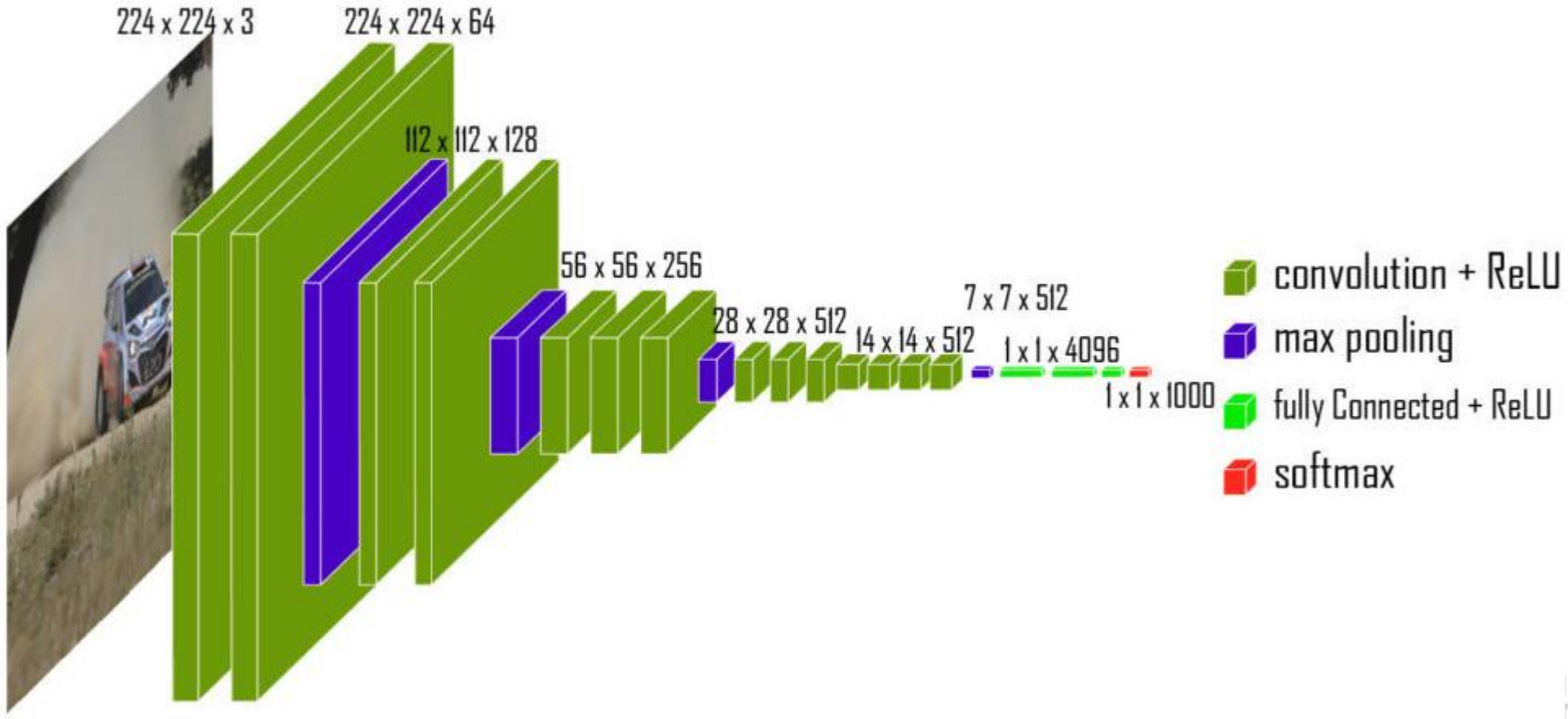
❑There are some changes proposed in this pre-trained network, These changes are:

1. The network is modified in such a way that it two inputs the image and list of region proposals generated on that image.
2. Second, the last pooling layer (here (7*7*512)) before fully connected layers needs to be replaced by the region of interest (RoI) pooling layer.
3. Third, the last fully connected layer and softmax layer is replaced by twin layers of softmax classifier and K+1 category-specific bounding box regressor with a fully connected layer.

Fast RCNN

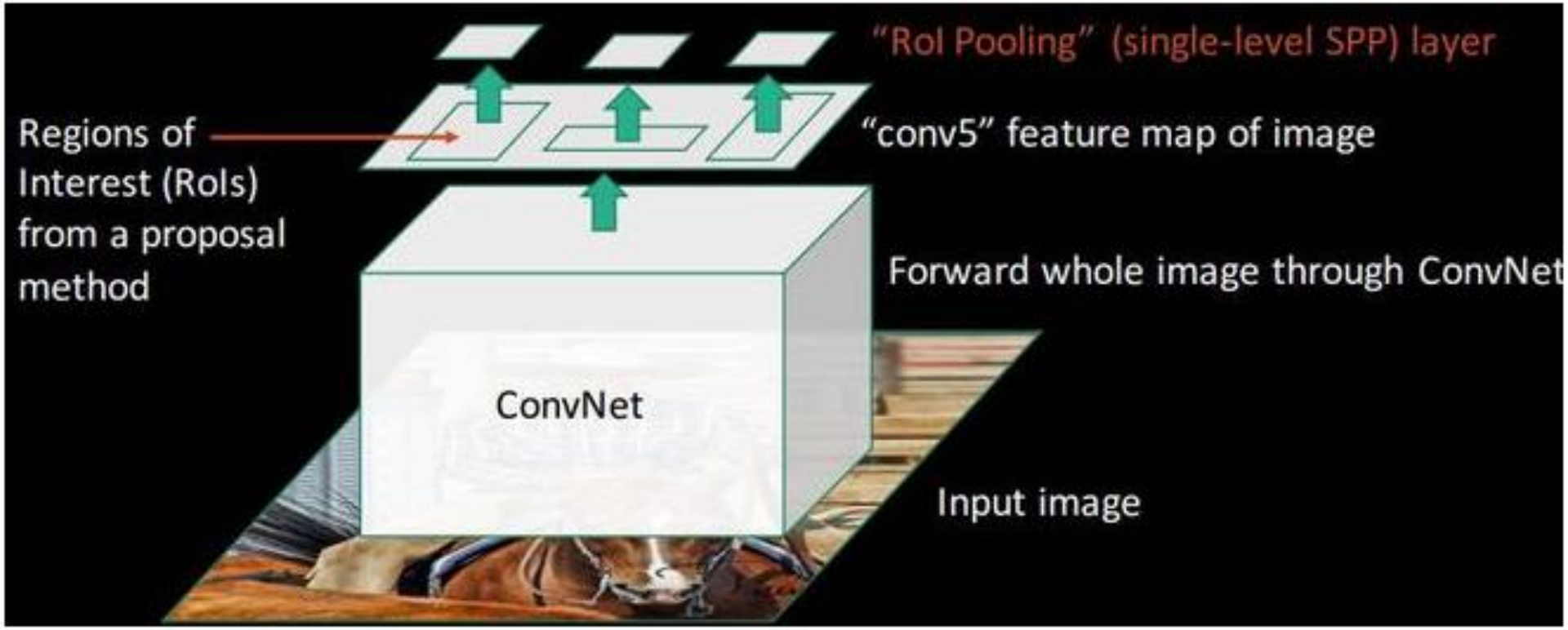
❑CNN Network of Fast R-CNN

❑This CNN architecture takes the image (size = 224 x 224 x 3 for VGG-16) and its region proposal and outputs the convolution feature map (size = 14 x 14 x 512 for VGG-16).



Fast RCNN

❑Region of Interest (RoI) pooling:



Fast RCNN

❑Region of Interest (RoI) pooling:

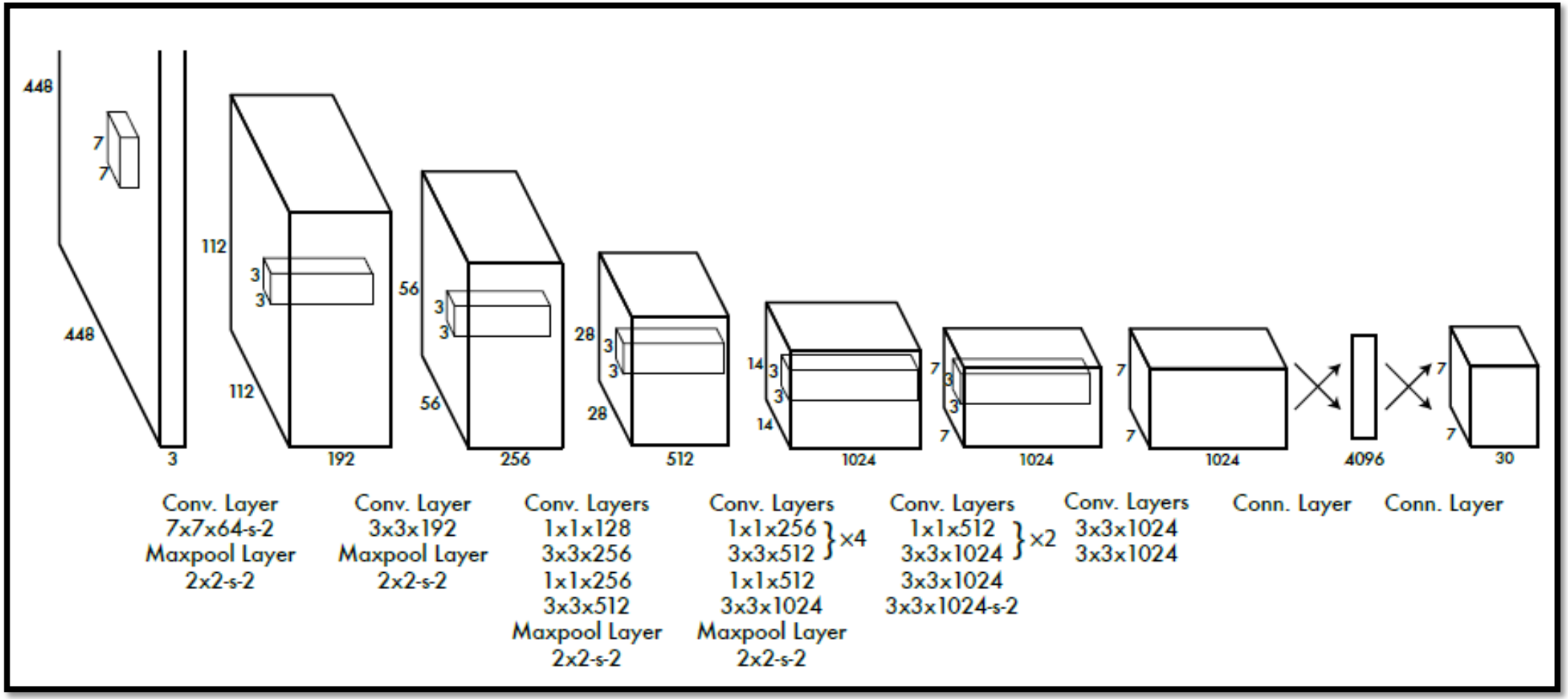
- ❑RoI pooling is a novel thing that was introduced in the Fast R-CNN paper.
- ❑Its purpose is to produce uniform, fixed-size feature maps from non-uniform inputs (RoIs).
- ❑It takes two values as inputs:

1. A feature map was obtained from the previous CNN layer (14 x 14 x 512 in VGG-16).
2. An N x 4 matrix represents regions of interest, where N is a number of RoIs, the first two represent the coordinates of the upper left corner of RoI and the other two represent the height and width of RoI denoted as (r, c, h, w).

You Only Look Once (YOLO): Real Time Object Detection

- **YOLO** was proposed by Joseph Redmond et al. in 2015.
- It was proposed to deal with the problems faced by the object recognition models at that time, Fast R-CNN is one of the state-of-the-art models at that time but it has its own challenges such as this network cannot be used in real-time, because it takes 2-3 seconds to predicts an image and therefore cannot be used in real-time.
- Whereas, in YOLO we have to look only once in the network i.e. only one forward pass is required through the network to make the final predictions.

You Only Look Once (YOLO): Real Time Object Detection



You Only Look Once (YOLO): Real Time Object Detection

- This architecture takes an image as input and resizes it to 448*448 by keeping the aspect ratio same and performing padding.
- This image is then passed in the CNN network. This model has 24 convolution layers, 4 max-pooling layers followed by 2 fully connected layers.
- For the reduction of the number of layers (Channels), we use 1*1 convolution that is followed by 3*3 convolution.
- Notice that the last layer of YOLOv1 predicts a cuboidal output. This is done by generating (1, 1470) from final fully connected layer and reshaping it to size (7, 7, 30).
- This architecture uses Leaky ReLU as its activation function in whole architecture except the last layer where it uses linear activation function.
- Batch normalization also helps to regularize the model. Dropout technique is also used to prevent overfitting.

You Only Look Once (YOLO): Real Time Object Detection

❑Training:

❑This model is trained on the ImageNet-1000 dataset.

❑The model is trained over a week and achieve top-5 accuracy of 88% on ImageNet 2012 validation which is comparable to GoogLeNet (2014 ILSVRC winner), the state of the art model at that time.

❑Fast YOLO uses fewer layers (9 instead of 24) and fewer filters. Except this, the fast YOLO have all parameters similar to YOLO.

❑YOLO uses sum-squared error loss function which is easy to optimize.

❑However, this function gives equal weight to the classification and localization task. The loss function defined in YOLO as follows:

You Only Look Once (YOLO): Real Time Object Detection

❑Training: Loss Function

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

where,

l_i^{obj} denotes if object is present in cell *i*.

l_{ij}^{obj} denotes j_{th} bounding box responsible for prediction of object in the cell *i*.

λ_{coord} and $lambda_{noobj}$ are regularization parameter required to balance the loss function.

In this model, we take $\lambda_{coord} = 5$ and $\lambda_{noobj} = .5$

You Only Look Once (YOLO): Real Time Object Detection

❑Training:

❑The first two parts of the above loss equation represent localization mean-squared error, but the other three parts represent classification error.

❑In the localization error, the first term calculates the deviation from the ground truth bounding box.

❑The second term calculates the square root of the difference between height and width of the bounding box.

❑In the second term, we take the square root of width and height because our loss function should be able to consider the deviation in terms of the size of the bounding box.

❑For small bounding boxes, the little deviation should be more important as compared to large bounding boxes.

❑There are three terms in classification loss, the first term calculates the sum-squared error between the predicted confidence score that whether the object present or not and the ground truth for each bounding box in each cell.

❑Similarly, The second term calculates the mean-squared sum of cells that do not contain any bounding box, and a regularization parameter is used to make this loss small.

❑The third term calculates the sum-squared error of the classes belongs to these grid cells.

You Only Look Once (YOLO): Real Time Object Detection

❑Detection:

❑This architecture divides the image into a grid of *S*S* size.

❑If the centre of the bounding box of the object is in that grid, then this grid is responsible for detecting that object.

❑Each grid predicts bounding boxes with their confidence score.

❑Each confidence score shows how accurate it is that the bounding box predicted contains an object and how precise it predicts the bounding box coordinates wrt. ground truth prediction.

You Only Look Once (YOLO): Real Time Object Detection

❑**Detection:**

❑At test time we multiply the conditional class probabilities and the individual box confidence predictions.

❑ We define our confidence score as follows :

$$P_r\left(Object\right) * IOU_{pred}^{truth}$$

❑Note, the confidence score should be *0* when there is no object exists in the grid.

❑If there is an object present in the image the confidence score should be equal to IoU between ground truth and predicted boxes.

❑Each bounding box consists of 5 predictions: *(x, y, w, h)* and confidence score.

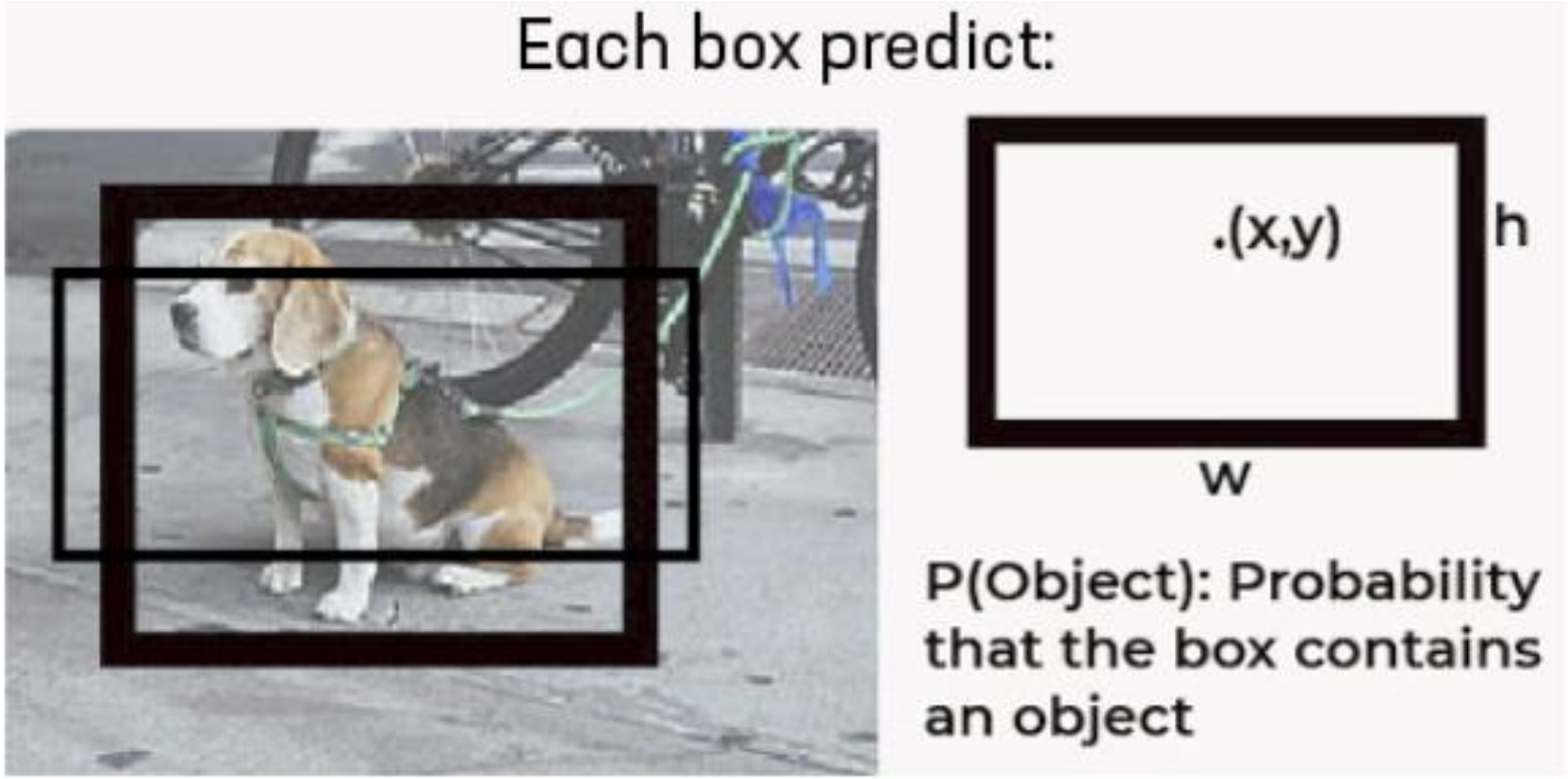
❑The *(x, y)* coordinates represent the centre of the box relative to the bounds of the grid cell.

❑The *h, w* coordinates represents height, width of bounding box relative to *(x, y)*.

❑The confidence score represents the presence of an object in the bounding box.

You Only Look Once (YOLO): Real Time Object Detection

❑**Detection:**



YOLO single Grid Bounding box-Box

You Only Look Once (YOLO): Real Time Object Detection

❑**Detection:**

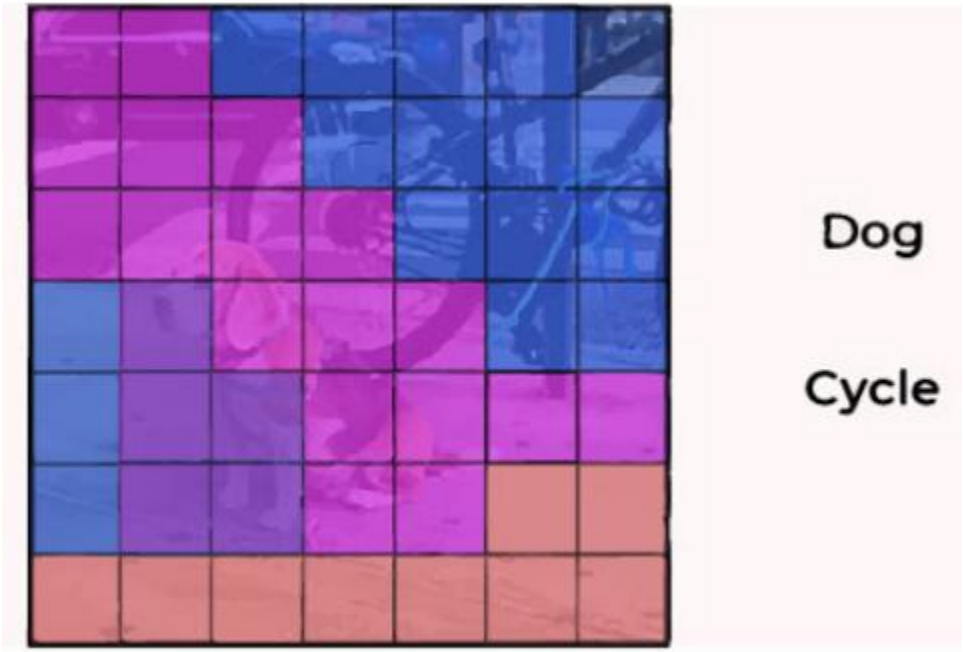
❑This results in combination of bounding boxes from each grid like this.

YOLO bounding box Combination

You Only Look Once (YOLO): Real Time Object Detection

❑Detection:

❑Each grid also predicts C conditional class probability, P_r ($Class_i$ | Object).



YOLO conditional probability map

You Only Look Once (YOLO): Real Time Object Detection

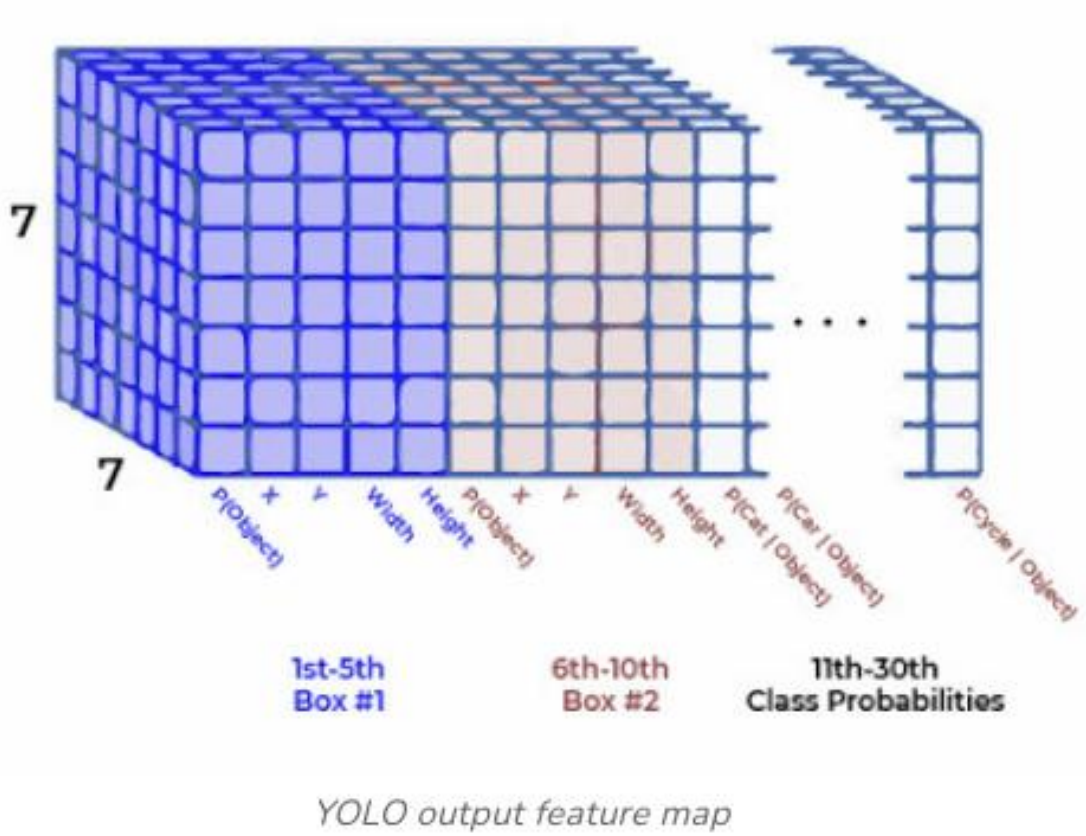
❑Detection:

❑This probability were conditional based on the presence of an object in grid cell.

❑Regardless the number of boxes each grid cell predicts only one set of class probabilities.

❑These prediction are encoded in the 3D tensor of size S * S * (5*B +C).

❑Now, we multiply the conditional class probabilities and the individual box confidence predictions.

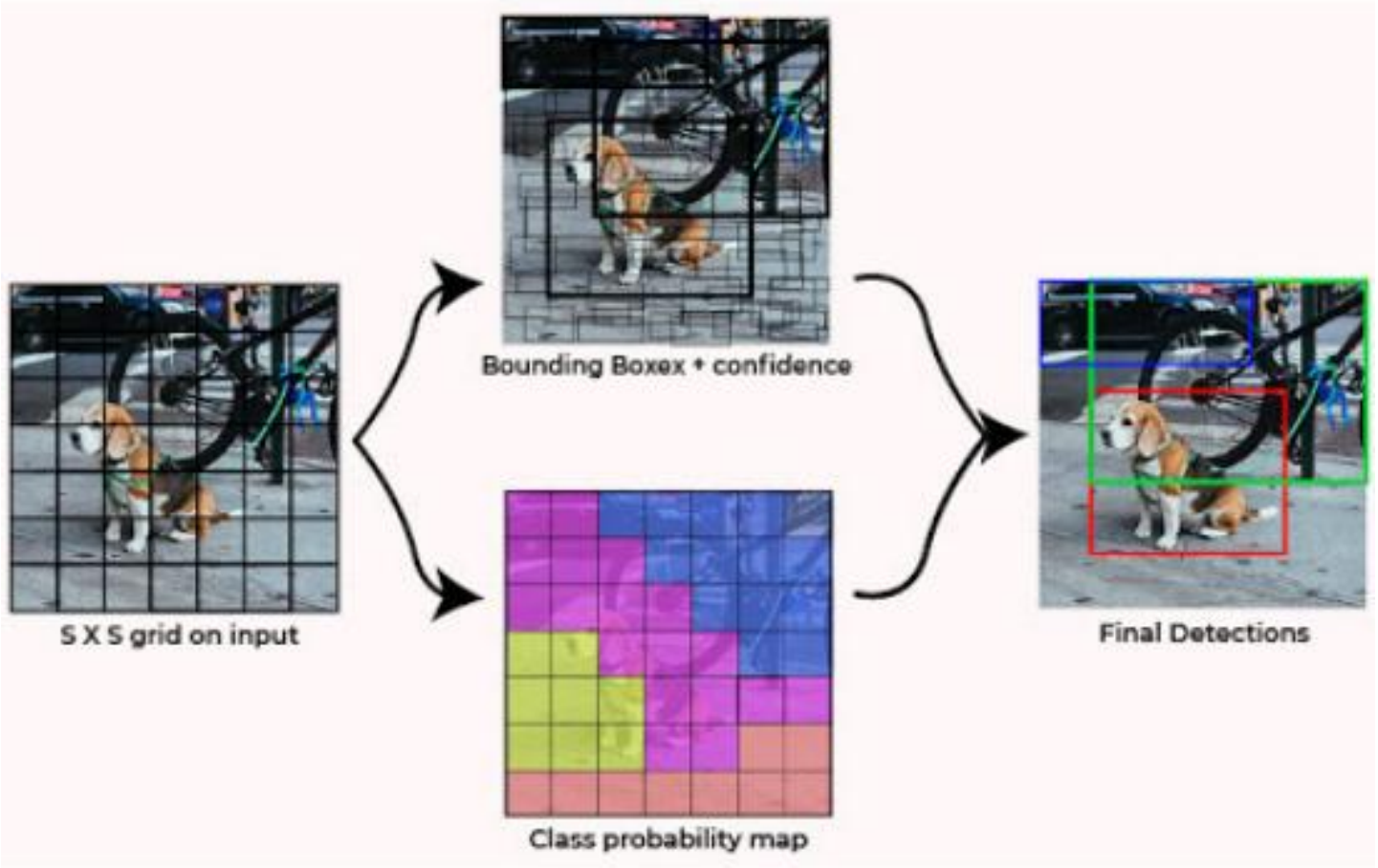


YOLO output feature map

You Only Look Once (YOLO): Real Time Object Detection

❑Detection:

$$P_r\left(Class_i|Object\right) * P_r\left(Object\right) * IOU_{pred}^{truth} = P_r\left(Class_i\right) * IOU_{pred}^{truth}$$



YOLO test Result

You Only Look Once (YOLO): Real Time Object Detection

❑**Detection:**

❑Which gives us class-specific confidence scores for each box.

❑These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

❑Then after we apply non-maximal suppression for suppressing the non max outputs(when a number of boxes are predicted for the same object).

❑And at last , our final predictions are generated.

❑YOLO is very fast at the test time because it uses only a single CNN architecture to predict results and class is defined in such a way that it treats classification as a regression problem.

You Only Look Once (YOLO): Real Time Object Detection

❑**Results:**

❑The simple YOLO has a mAP (mean average precision) of 63.4% when trained on VOC in 2007 and 2012, the Fast YOLO which is almost 3x faster in result generation has mAP of 52%.

❑This is lower than the best Fast R-CNN model achieved (71% mAP) and also the R-CNN achieved (66% mAP).

❑However, it beats other real-time detectors such as (DPMv5 33% mAP) on accuracy.

You Only Look Once (YOLO): Real Time Object Detection

❑Benefits of YOLO:

❑Process frames at the rate of *45 fps* (larger network) to *150 fps*(smaller network) which is better than real-time.

❑The network is able to generalize the image better.

❑Disadvantages of YOLO:

❑Comparatively low recall and more localization error compared to Faster R_CNN.

❑Struggles to detect close objects because each grid can propose only 2 bounding boxes.

❑Struggles to detect small objects.

Single Shot Detector

❑How single-shot detector (SSD) works?

❑Object detection is a critical task in computer vision, with applications ranging from autonomous driving to image retrieval and surveillance.

❑The Single Shot Detector (SSD) is an advanced algorithm that has revolutionized this field by enabling real-time detection of objects in images.

Single Shot Detector

❑Introduction to Single-shot Detector

❑Object detection involves identifying and locating objects within an image.

❑Traditional methods required multiple passes over the image, making them computationally expensive and slow. SSD simplifies this process by detecting objects in a single pass, hence the name "*Single Shot Detector*."

❑This approach not only speeds up the detection process but also maintains high accuracy, making SSD a popular choice for real-time applications.

Single Shot Detector

❑Model Architecture

1. **Base Network:** The SSD architecture begins with a pre-trained convolutional neural network (CNN) known as the base network. Commonly, networks like VGG16 are used due to their strong feature extraction capabilities. The base network processes the input image and generates feature maps, which are essential for object detection.
2. **Extra Layers:** Beyond the base network, SSD includes extra convolutional layers. These layers progressively decrease in size and are responsible for detecting objects at different scales. Each additional layer generates feature maps that contribute to the final detection process.

Single Shot Detector

❑Feature Maps and Multi-scale Detection

❑A standout feature of SSD is its use of multi-scale feature maps.

❑These maps capture information at various resolutions, allowing SSD to detect objects of different sizes effectively.

❑Higher resolution feature maps are adept at detecting smaller objects, while lower resolution maps handle larger objects.

Single Shot Detector

❑Default Boxes (Anchor Boxes)

❑SSD employs a technique called default boxes (also known as anchor boxes) at each location in the feature maps.

❑These boxes come in various aspect ratios and scales, providing a diverse set of potential object locations.

❑Each default box is associated with two sets of predictions:

1. **Class Scores**: These scores indicate the likelihood of an object belonging to a specific class.
2. **Bounding Box Offsets**: These offsets refine the default box to better match the actual object's location.

Single Shot Detector

- **Predictions:** For each default box, SSD predicts:
- **Class Confidences**: The probability of the box containing a specific object class.
- **Bounding Box Adjustments**: The coordinates to refine the position and size of the default box to match the detected object more precisely.

Single Shot Detector

- **Loss Function:** The SSD loss function combines two components:
- **Localization Loss (Lloc)**: This measures how accurately the predicted bounding boxes match the ground truth boxes using Smooth L1 loss.
- **Confidence Loss (Lconf)**: This evaluates the confidence in the predicted class scores using softmax loss.

Single Shot Detector

❑Non-Maximum Suppression (NMS):

❑To finalize the detection process, SSD applies Non-Maximum Suppression (NMS).

❑This step eliminates redundant boxes with lower confidence scores, ensuring that only the most confident and relevant predictions are retained.

Single Shot Detector

❑Steps in Single Shot Detection

1. **Input Image**: The image is passed through the base network to extract feature maps.
2. **Feature Extraction**: The extra layers process these maps at multiple scales.
3. **Default Boxes Assignment**: Default boxes of various sizes and aspect ratios are assigned to each feature map cell.
4. **Prediction**: For each default box, class scores and bounding box offsets are predicted.
5. **Loss Calculation**: The loss is computed based on localization and confidence.
6. **NMS**: Redundant boxes are removed to produce the final set of detections.

Note for Students

- This power point presentation is for lecture, therefore it is suggested that also utilize the text books and lecture notes.