COURSE: DSA4050

COURSE TITLE: Deep Learning for Computer Vision

INSTRUCTOR: DR. EDWARD OMBUI

Week 4 Lesson Notes: Advanced CNN Architectures and Transfer Learning

---

Overview

This week, we will explore **advanced Convolutional Neural Network (CNN) architectures** that have significantly improved performance in **image recognition and classification**. You will learn about three powerful CNN architectures:

- **Inception:** Uses multi-scale convolution filters to improve feature extraction while reducing computational complexity.
- **ResNet (Residual Networks):** Introduces residual connections to solve the **vanishing gradient problem**, making it easier to train very deep networks.
- **DenseNet (Densely Connected Networks):** Enhances information flow by creating shorter connections between layers.

You will compare these architectures, analyze their strengths and weaknesses, and determine when each is most appropriate for specific tasks.

Additionally, you will be introduced to **Transfer Learning**, a technique that enables you to **leverage pre-trained models** for new tasks. Transfer learning is especially useful when:

- You have **limited data** for training.
- Training a model from scratch is **computationally expensive**.
- You want to **adapt a pre-trained model (e.g., trained on ImageNet) to a new dataset**.

Finally, you will **implement and train an advanced CNN architecture on a new dataset**, gaining hands-on experience in **fine-tuning a pre-trained model** for a specialized task.

Advanced CNN Architectures

1. Inception (GoogLeNet)

📌 **Key Idea:** Instead of choosing a single convolution size (3×3, 5×5, etc.), **Inception networks use multiple filter sizes in parallel**, extracting features at different scales.

✅ **Advantages:**

- More efficient than stacking deep layers.
- Captures fine and coarse details simultaneously.
- Reduces computational cost with **1×1 convolutions** (bottleneck layers).

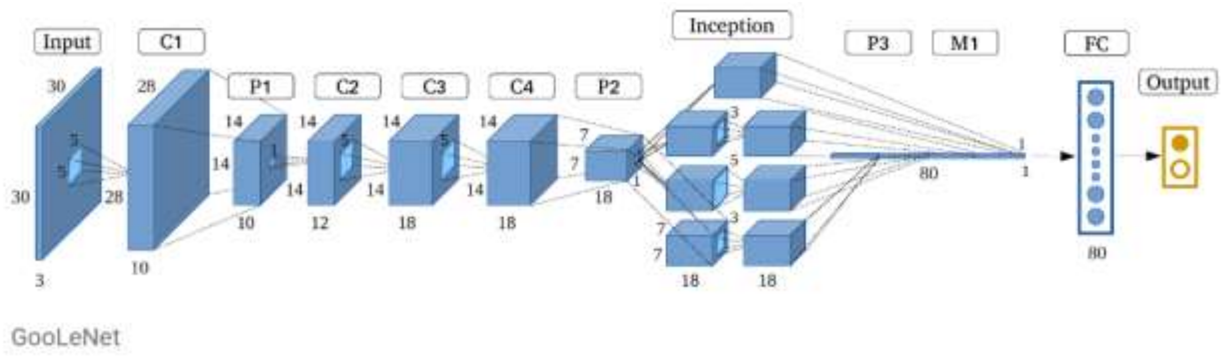
❌ **Disadvantages:**

- More complex than traditional CNNs.
- Requires careful tuning of hyperparameters.

Architecture Diagram:

Source : https://viso.ai/deep-learning/googlenet-explained-the-inception-model-that-won-imagenet/

.



Implementation Example (Using TensorFlow):

```
python
CopyEdit
from tensorflow.keras.applications import InceptionV3
model = InceptionV3(weights='imagenet', include_top=False, input_shape=(224,
224, 3))
```

2. ResNet (Residual Networks)

📌 **Key Idea: Solves the vanishing gradient problem** by introducing **residual connections (skip connections)**, allowing gradients to flow more easily through deep networks.

✅ **Advantages:**

- Enables training of very **deep networks (e.g., 50, 101, 152 layers)**.
- Residual connections improve convergence.
- Better accuracy than traditional deep CNNs.

❌ **Disadvantages:**

- Increased computational cost.
- Requires careful design of residual blocks.

Architecture Diagram:
source: https://medium.com/towards-data-science/residual-blocks-in-deep-learning-11d95ca12b00

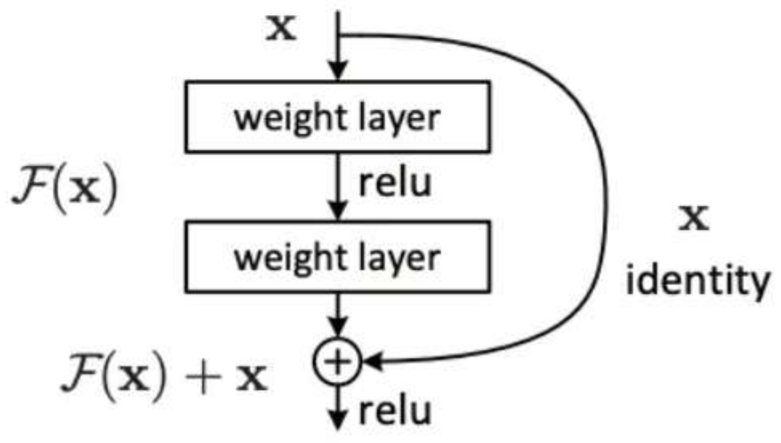

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 3: Residual block with skip connection (Image by Kaiming)

Implementation Example (Using TensorFlow):

```
python
CopyEdit
from tensorflow.keras.applications import ResNet50
```

```
model = ResNet50(weights='imagenet', include_top=False, input_shape=(224,
224, 3))
```

---

3. DenseNet (Densely Connected Networks)

📌 **Key Idea:** Unlike traditional CNNs where layers pass information sequentially, **DenseNet connects each layer to every other layer**, improving feature reuse and gradient flow.

✅ **Advantages:**

- **Better information flow** across layers.
- **Reduces overfitting** due to feature reuse.
- **Requires fewer parameters** compared to ResNet.

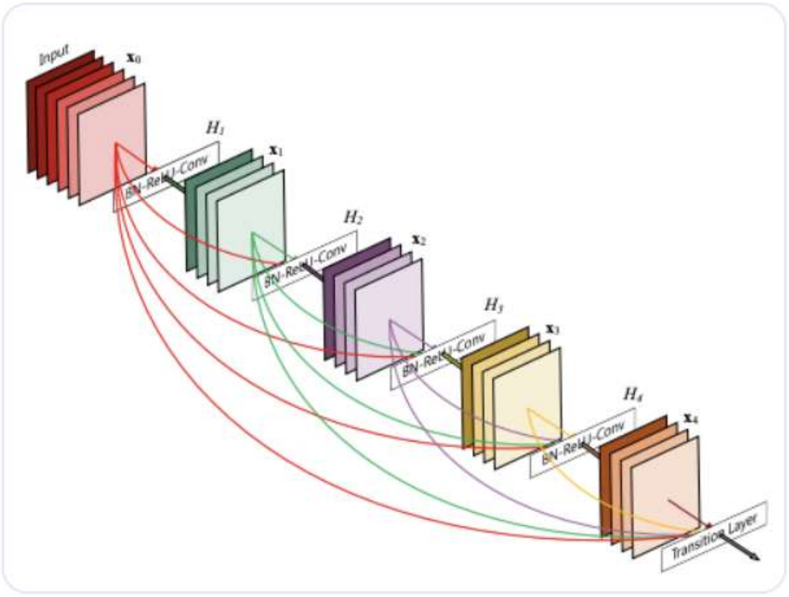
❌ **Disadvantages:**

- Increased memory usage due to dense connections.
- More complex computation during training.

Architecture Diagram:

Source: https://www.digitalocean.com/community/tutorials/popular-deep-learning-architectures-densenet-mnasnet-shufflenet

As the layers in the network receive feature maps from all the preceding layers, the network will be thinner and more compact. Below is a 5-layer dense block with the number of channels set to 4.



The DenseNet Architecture

Implementation Example (Using TensorFlow):

```
python
CopyEdit
from tensorflow.keras.applications import DenseNet121
model = DenseNet121(weights='imagenet', include_top=False, input_shape=(224,
224, 3))
```

Transfer Learning: Leveraging Pre-Trained Models

What is Transfer Learning?

🚀 Transfer learning allows us to **use a model pre-trained on a large dataset (e.g., ImageNet) for a new task** with minimal training. Instead of training a model from scratch, we:

1. Use a **pre-trained model** as a feature extractor.
2. Fine-tune the model by replacing and training the last few layers for the new task.

✅ **Advantages:**

- Requires **less data** to train.
- Speeds up training.
- Leverages **knowledge from previous large-scale training**.

Implementation Steps:

1 Load a pre-trained model (without the top classification layer).

```
python
CopyEdit
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten

# Load pre-trained model
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,
224, 3))

# Add custom classification layers
x = Flatten()(base_model.output)
x = Dense(256, activation='relu')(x)
x = Dense(10, activation='softmax')(x)  # 10-class classification

# Define new model
model = Model(inputs=base_model.input, outputs=x)
```

2 Freeze pre-trained layers (optional) to retain learned features.

```
python
CopyEdit
for layer in base_model.layers:
    layer.trainable = False  # Freeze layers
```

3 Compile and Train the Model

```
python
CopyEdit
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, validation_data=(val_images,
val_labels))
```

Comparing CNN Architectures

Architecture	Key Innovation	Strengths	Best For
Inception	Multi-scale feature extraction	Efficient, captures details at different levels	Object detection, classification
ResNet	Residual (skip) connections	Enables very deep networks, solves vanishing gradient problem	Very deep models, image recognition
DenseNet	Dense connections between layers	Feature reuse, improved gradient flow	Image classification, segmentation

Learning Outcomes

By the end of this session, you should be able to:
✅ **Compare advanced CNN architectures** (Inception, ResNet, DenseNet) and their use cases.
✅ **Apply transfer learning** to fine-tune a pre-trained model for a new task.
✅ **Implement and train an advanced CNN model** on a new dataset.

Activities

1. Practical Exercise: Transfer Learning Implementation

💡 **Task:** Fine-tune a **pre-trained ResNet50 model** on a new dataset (e.g., Cats vs. Dogs).

Steps:

1. **Load and preprocess dataset** (resize images, normalize pixel values).
2. **Load ResNet50 model** without the top layer.
3. **Add new classification layers** for the target dataset.
4. **Freeze the base layers** and train the model.
5. **Evaluate accuracy** and visualize results.

📌 **Bonus Challenge:** Try fine-tuning **Inception or DenseNet** on a custom dataset.

Conclusion

This lesson covered:
✅ **Three advanced CNN architectures** (Inception, ResNet, DenseNet).
✅ **How transfer learning works** and its benefits.
✅ **Hands-on implementation of transfer learning** with a pre-trained model.

Next Steps:
🔹 Experiment with different architectures for your project.
🔹 Try applying transfer learning to a dataset relevant to your domain.
🔹 Explore **real-world applications** of advanced CNNs (e.g., medical image analysis, self-driving cars).