Pre-Trained Models

- **Image Classification**: Some popular pre-trained models for image classification include VGG-16, Inception, ResNet50, and EfficientNet[1][2][3]

- **Object Detection**: Pre-trained models for object detection include Faster R-CNN with ResNet, Faster R-CNN with MobileNet, and RetinaNet[4].

- **Face Detection**: Pre-trained models for face detection include MTCNN and InceptionResnet[5][6].

- **Text Detection**: Pre-trained models for text detection include XLNet, ERNIE, Text-to-Text Transfer Transformer (T5), Binary Partitioning Transfomer (BPT), Neural Attentive Bag-of-Entities (NABoE), and Rethinking Complex Neural Network Architectures[7].

Using pre-trained models in your project typically involves the following steps:

1. **Identify the Right Model Architecture**: Depending on your task, you need to choose the right model architecture. For example, VGG-16 or ResNet50 for image classification, Faster R-CNN for object detection, etc[1].

2. **Load the Pretrained Weights**: Once you've chosen the model, you need to load the pretrained weights for that model. This can be done using functions provided by your deep learning framework. For example, in PyTorch, you can use the `torch.load()` function[1].

3. **Fine-Tuning**: In many cases, you might need to fine-tune the pretrained model on your specific task. This involves unfreezing the top layers of the model and continuing the training[2]. Remember to compile the model before fine-tuning[2].

4. **Make Predictions**: Once the model is fine-tuned, you can use it to make predictions on new data[3].