# 🎬 VideoX Action Recognition - Complete Setup Guide

## 📋 Table of Contents



---

## 🔧 Prerequisites

**Required:**

- Windows 10/11
- Python 3.8 or higher
- NVIDIA GPU with 12GB+ VRAM
- CUDA 11.8 or higher
- 20GB free disk space

**Check Your System:**

```bash
# Check Python
python --version
# Should show: Python 3.8 or higher

# Check CUDA
nvidia-smi
# Should show GPU info

# Check GPU memory
nvidia-smi --query-gpu=memory.total --format=csv
# Should show: 12GB or more
```

---

## 🚀 Installation Steps

### Step 1: Run Auto Installer

```powershell
# Save INSTALL_VIDEOX.ps1 to your project folder
# Then run:

.\INSTALL_VIDEOX.ps1

# Or specify custom path:
.\INSTALL_VIDEOX.ps1 -ProjectPath "D:\MyProject"
```

**What it does:**

- ✅ Creates virtual environment
- ✅ Installs PyTorch with CUDA
- ✅ Installs VideoX and dependencies
- ✅ Downloads NLTK data
- ✅ Creates project structure
- ✅ Downloads pre-trained models
- ✅ Verifies installation

**Time required:** 15-30 minutes

---

### Step 2: Copy Project Files

After installation, copy these files to your project:

**Core Files:**

1. **src/model_architecture.py** - VideoX model (provided above)
2. **src/text_processor.py** - Updated text processor
3. **src/data_preparation.py** - Dataset preparation
4. **src/dataset_loader.py** - Video loading
5. **src/trainer_module.py** - Training loop
6. **src/inference_module.py** - Inference engine

7. **src/evaluator.py** - Evaluation metrics
8. **src/vocabulary_builder.py** - Vocabulary management
9. **src/__init__.py** - Package init

**Application Files:**

10. **main.py** - Main entry point
11. **app.py** - Flask API server
12. **web_interface.html** - Web UI
13. **api_client.py** - Python client

**Configuration:**

14. **config/config.yaml** - Main configuration
15. **requirements.txt** - Python dependencies

**Documentation:**

16. **README.md** - Project overview
17. **ANNOTATION_GUIDE.md** - Annotation guidelines
18. **API_USAGE_GUIDE.md** - API documentation

---

## 📁 Project Structure

After installation, your project should look like:

```
E:\OCR_system-Atlas\
│
├── VideoX\                  # VideoX repository (cloned)
├── venv\                    # Virtual environment
│
├── config\
│   └── config.yaml          # Main configuration
│
├── src\
│   ├── __init__.py
│   ├── model_architecture.py  # VideoX model
│   ├── text_processor.py      # Text processing
│   ├── data_preparation.py    # Data prep
│   ├── dataset_loader.py      # Video loading
│   ├── trainer_module.py      # Training
│   ├── inference_module.py    # Inference
│   ├── evaluator.py           # Evaluation
│   └── vocabulary_builder.py  # Vocabulary
│
├── data\
│   ├── videos\              # PUT YOUR MP4 FILES HERE
│   ├── annotations\         # Generated annotations
│   └── splits\              # Train/val splits
│
├── models\
│   └── videox\              # Pre-trained VideoX models
│
├── checkpoints\             # Trained model checkpoints
│
├── outputs\
│   ├── predictions\         # Prediction results
│   └── evaluations\         # Evaluation results
│
├── uploads\                 # Uploaded videos (API)
├── api_results\             # API client results
├── logs\                    # Training logs
│
├── main.py                  # Main script
├── app.py                   # API server
├── web_interface.html       # Web UI
├── api_client.py            # Python client
│
├── requirements.txt         # Dependencies
```

```
├── README.md          # Documentation
└── INSTALL_VIDEOX.ps1 # Installer script
```

---

## ⚙️ Configuration

### Update `config/config.yaml`

```yaml
```

```yaml
# Model Configuration
model:
  # VideoX model (or CLIP as fallback)
  videox_model: "microsoft/videox-base"
  clip_model: "openai/clip-vit-base-patch32"

  # Model architecture
  d_model: 768
  temporal_layers: 4
  num_frames: 16
  frame_size: [224, 224]
  num_classes: 50
  dropout: 0.1

# Training Configuration
training:
  num_epochs: 50
  batch_size: 1
  gradient_accumulation_steps: 8
  learning_rate: 0.00005  # Lower for VideoX
  weight_decay: 0.01
  use_fp16: true
  checkpoint_dir: "checkpoints"
  save_every: 10

  # VideoX-specific
  freeze_backbone: true
  unfreeze_after_epoch: 20

# Easy Mode Rules
easy_mode:
  min_duration: 8
  max_duration: 40
  max_words: 25
  use_ing_verbs: true
  allow_the: true
  forbidden_words: ["next", "other", "carefully", "inspect"]
  goal_oriented: true

# Inference Configuration
inference:
  confidence_threshold: 0.5
  boundary_start_threshold: 0.5
```

```
  boundary_end_threshold: 0.5
  min_action_duration: 8.0
  max_action_duration: 40.0
  generate_captions: true

# Add your video annotations here
annotations_raw:
  f1: |
    0:00.0-0:20.0#1 Assembling black ballpoint pens
    0:20.0-0:51.0#2 Assembling blue ballpoint pens
```

---

## 🎯 Usage

### 1. Prepare Data

```
bash

# Activate environment
venv\Scripts\activate

# Add your videos to data\videos\

# Prepare dataset
python main.py --mode prepare
```

### 2. Train Model

```
bash

# Train for 50 epochs
python main.py --mode train --epochs 50

# Or quick test (10 epochs)
python main.py --mode train --epochs 10

# Resume from checkpoint
python main.py --mode train --resume
```

### 3. Make Predictions

```
bash
```

```
# Predict all videos
python main.py --mode predict

# Predict single video
python main.py --mode predict --video data/videos/f1.mp4
```

### 4. Evaluate

```
bash

python main.py --mode evaluate
```

### 5. Run Everything

```
bash

# Run all stages
python main.py --mode all
```

---

## 🌐 API Usage

### Start API Server

```
bash

python app.py
```

Open browser: **http://localhost:5000**

### Use Python Client

```
python
```

```
from api_client import ActionRecognitionClient

client = ActionRecognitionClient("http://localhost:5000")

# Check status
status = client.check_status()
print(status)

# Annotate video
result = client.annotate_video("video.mp4")
print(f"Found {result['num_segments']} segments")

# Batch process
videos = ["video1.mp4", "video2.mp4", "video3.mp4"]
results = client.annotate_videos(videos)
```

### Use REST API

```
bash

# Upload video
curl -X POST http://localhost:5000/api/annotate \
    -F "video=@video.mp4"

# Get results
curl http://localhost:5000/api/results/f1

# Download Atlas format
curl http://localhost:5000/api/download/f1/atlas
```

---

## 🐛 Troubleshooting

### Problem 1: VideoX Not Found

**Error:** `ImportError: No module named 'videox'`

**Solution:**

```
bash
```

```
cd VideoX
pip install -e .
cd ..
```

### Problem 2: CUDA Out of Memory

**Error:** `RuntimeError: CUDA out of memory`

**Solution:** Edit `config/config.yaml`:

```yaml
training:
  batch_size: 1
  gradient_accumulation_steps: 16  # Increase
  use_fp16: true
```

### Problem 3: Model Download Fails

**Error:** `Failed to download model`

**Solution:**

```bash
# Manual download
pip install huggingface-hub
python -c "from huggingface_hub import hf_hub_download; \
hf_hub_download(repo_id='microsoft/videox-base', \
filename='pytorch_model.bin', local_dir='models/videox')"
```

### Problem 4: Import Errors

**Error:** `ModuleNotFoundError`

**Solution:**

```bash
pip install -r requirements.txt --upgrade
```

### Problem 5: Video Loading Fails

**Error:** `Failed to load video`

**Solution:**

- Check video format (MP4 recommended)
- Check OpenCV: `pip install opencv-python-headless --upgrade`
- Try re-encoding: `ffmpeg -i input.mp4 -c:v libx264 output.mp4`

**Problem 6: Training Too Slow**

**Solution:**

- Check GPU usage: `nvidia-smi`
- Verify CUDA: `python -c "import torch; print(torch.cuda.is_available())"`
- Reduce num_frames: 16 → 8
- Enable FP16: `use_fp16: true`

---

## 📊 Performance Tips

**For Faster Training:**

```yaml
model:
  num_frames: 8  # Reduce from 16
  temporal_layers: 2  # Reduce from 4

training:
  batch_size: 2  # Increase if memory allows
  use_fp16: true
  freeze_backbone: true  # Don't train VideoX
```

**For Better Accuracy:**

```yaml
```

```yaml
model:
  num_frames: 32  # Increase
  temporal_layers: 6  # Increase

training:
  num_epochs: 100  # More epochs
  unfreeze_after_epoch: 30  # Fine-tune VideoX
```

---

## 🎓 Next Steps

**Week 1:**

- [ ] Complete installation
- [ ] Test on 1 video
- [ ] Verify model works

**Week 2-4:**

- [ ] Annotate 20 videos
- [ ] Train model
- [ ] Test API

**Month 2-3:**

- [ ] Annotate 50+ videos
- [ ] Fine-tune VideoX
- [ ] Deploy to production

---

## 📚 Additional Resources

- VideoX GitHub
- ANNOTATION_GUIDE.md
- API_USAGE_GUIDE.md
- Troubleshooting Wiki

---

## ✅ Verification Checklist

Before starting, verify:

- [ ] Python 3.8+ installed
- [ ] NVIDIA GPU (12GB+ VRAM)
- [ ] CUDA 11.8+ installed
- [ ] 20GB free disk space
- [ ] All files copied correctly
- [ ] Virtual environment created
- [ ] Dependencies installed
- [ ] VideoX imported successfully
- [ ] Config file updated
- [ ] Videos in data/videos/

---

## 🆘 Getting Help

If you encounter issues:

1. Check this guide
2. Review error messages carefully
3. Check troubleshooting section
4. Verify all dependencies installed
5. Test with provided examples

---

**Ready to start!** 🚀

Run: `.\INSTALL_VIDEOX.ps1` and let's build something amazing!