NanoDet Custom Training Documentation

# 1. Introduction

This document outlines the complete process for setting up, modifying, and training a customized NanoDet object detection model.   
It includes installation steps, version information, resolved errors, all Python source code modifications, configuration adjustments, and file paths.

# 2. Environment Setup

- Python Version: 3.10  
- PyTorch Version: Compatible with CUDA 11.7+  
- Google Colab: Used for final training due to GPU constraints on local machine.

# 3. Installation Steps

1. Clone NanoDet repository (on local, not on Colab).  
2. Modify NanoDet code as needed.  
3. Zip `nanodet\_folder/` and `nanodet\_dataset/`.  
4. Upload to Google Colab and unzip using the following commands:

```python  
!unzip nanodet\_project.zip -d /content/  
!unzip dataset.zip -d /content/  
```

# 4. Python Dependencies

Install required packages in Colab:

```bash  
pip install -r requirements.txt  
# or manually install  
pip install torch torchvision torchaudio pytorch-lightning onnx onnxsim opencv-python  
```

# 5. Configuration (YAML)

Ensure correct input channels in `config.yaml`:

```yaml  
head:  
 input\_channel: [116, 232, 464]  
```

Update dataset paths to absolute Colab paths:

```yaml  
ann\_path: /content/nanodet\_dataset/train/\_annotations.coco.json  
img\_path: /content/nanodet\_dataset/train  
```

# 6. Python File Modifications

- File: `nanodet/model/head/nanodet\_plus\_head.py`

Update `\_\_init\_\_` to handle list `input\_channel`:

```python  
if isinstance(input\_channel, int):  
 self.in\_channels = [input\_channel] \* len(strides)  
else:  
 self.in\_channels = input\_channel  
```

Updated `\_init\_layers` to loop per input channel:

```python  
self.cls\_convs = nn.ModuleList()  
for in\_c in self.in\_channels:  
 cls\_convs = self.\_build\_not\_shared\_head(in\_c)  
 self.cls\_convs.append(cls\_convs)  
```

Updated `forward()` to apply all stacked convs per feature map:

```python  
for i, feat in enumerate(feats):  
 for conv in self.cls\_convs[i]:  
 feat = conv(feat)  
 output = self.gfl\_cls[i](feat)  
 outputs.append(output.flatten(start\_dim=2))  
```

# 7. Errors and Solutions

* ❌ Error: Conv2D input mismatch - got 192 but expected 96

✔️ Fix: Pass `input\_channel: [116, 232, 464]` in head config.

* ❌ Error: `TypeError: 'int' object is not iterable`

✔️ Fix: Check `input\_channel` type and convert to list if needed.

* ❌ Error: `export\_onnx.py` CLI flags not recognized

✔️ Fix: Match argument names: `--cfg\_path`, `--model\_path`, `--out\_path`, `--input\_shape`

# 8. Export to ONNX

Use this one-liner to export model:

```bash  
python tools/export\_onnx.py --cfg\_path config.yaml --model\_path workspace/model\_best/model\_best.pth --out\_path nanodet\_plus.onnx --input\_shape 320,320  
```

# 9. Training on Colab

Command to start training:

```bash  
python tools/train.py --cfg\_path config.yaml  
```

# 10. Timestamp

Generated on 2025-08-07 10:33:16

# 🔧 Additional Fixes: Imports and Dependency Issues

**Problem:**  
Some import statements in NanoDet codebase refer to incorrect or outdated module paths, which causes `ImportError` or `ModuleNotFoundError` during runtime.  
These mostly happen when using newer versions of PyTorch Lightning or certain project refactors.  
**Fix:**  
Update the import paths to their current and compatible locations with the installed `pytorch\_lightning` version.  
Here are a few corrected import lines:

from pytorch\_lightning.loggers.logger import LightningLoggerBase  
from pytorch\_lightning.loggers.logger import rank\_zero\_experiment  
from lightning\_utilities.core.rank\_zero import rank\_zero\_only  
from lightning\_utilities.core.imports import get\_filesystem  
from termcolor import colored

⚠️ These changes should be made in any custom logging files or utilities (e.g., `nanodet/util/logger.py`).