Dataset Cleaning Guide

Overview

Comprehensive guide for cleaning the basketball shooting training dataset to remove inappropriate images and ensure high-quality data for model training.

Created: December 13, 2025
Dataset Size: 19,451 images (before cleaning)

🎯 Cleaning Objectives

Images to KEEP ✅

1. **Individual player** - One person only
2. **Full body visible** - Ankles visible, not cut off
3. **Clear shooting motion** - Hands elevated near head level
4. **High visibility** - Person clearly visible (>60% landmark visibility)
5. **Adequate size** - Minimum 200x200 pixels
6. **Basketball context** - Player actually shooting basketball

Images to REMOVE ❌

1. **No person detected** - Empty courts, equipment only
2. **Multiple players** - Team photos, group shots
3. **Partial body** - Cropped images missing legs/feet
4. **Non-shooting poses** - Dribbling, passing, defending
5. **Non-basketball** - Soccer, swimming, other sports
6. **Low quality** - Blurry, too small, corrupted files
7. **Court scenes** - Stadium views, spectators

🔍 Current Dataset Issues Discovered

During initial assessment, we found:

Major Issues

- **Wrong Sport:** Soccer/football players in “basketball” dataset
- **Underwater Photos:** Swimming/diving images incorrectly labeled
- **Multiple Players:** Team photos instead of individual shots
- **No Basketball:** General sports photos without basketballs

Statistics from Sample Review

```
Sample Size: 30 images
✓ Valid Basketball Shooting: 8 (27%)
✗ Wrong Sport: 6 (20%)
✗ Multiple Players: 7 (23%)
✗ Non-Shooting Pose: 5 (17%)
✗ Partial Body: 4 (13%)
```

Estimated Removal Rate: ~70-75% of images will be removed

🛠️ Cleaning Tool: `clean_dataset.py`

Features

1. **Automated Detection:**
   - Uses MediaPipe Pose for person detection
   - Analyzes body visibility and pose
   - Checks image quality and dimensions

2. **Smart Filtering:**
   - Validates shooting pose (hands elevated)
   - Ensures full body visibility (ankles detected)
   - Measures landmark visibility scores

3. **Safe Quarantine:**
   - Moves (not deletes) rejected images
   - Preserves original filenames with reason codes
   - Allows manual review and recovery

4. **Detailed Reporting:**
   - JSON and Markdown reports
   - Statistics by directory
   - Removal reason breakdown

🚀 Usage

Basic Usage

```
cd /home/ubuntu/basketball_app/training_data
python3 clean_dataset.py
```

Interactive prompts:

```
BASKETBALL SHOOTING DATASET CLEANER
====================================
Base Directory: /home/ubuntu/basketball_app/training_data

This script will:
  1. Scan all images in training_data/
  2. Keep ONLY individual players in shooting poses
  3. Remove: multiple players, court scenes, non-shooting poses
  4. Move removed images to quarantine/ folder
  5. Generate detailed cleanup report

Proceed with cleanup? (yes/no): yes
```

Advanced Usage - Specific Directory

```python
from clean_dataset import DatasetCleaner

cleaner = DatasetCleaner("/path/to/dataset")
cleaner.clean_directory("form_quality_classifier/good_form")
cleaner.generate_report()
```

Custom Confidence Threshold

```python
# Modify in clean_dataset.py
pose = mp_pose.Pose(
    static_image_mode=True,
    model_complexity=2,
    enable_segmentation=False,
    min_detection_confidence=0.7  # Increase to 0.7 for stricter filtering
)
```

📊 Cleaning Algorithm

Step 1: Image Validation

```python
def analyze_image(image_path):
    # Check readability
    if not readable:
        return False, "unreadable"

    # Check dimensions
    if height < 200 or width < 200:
        return False, "too_small"

    # Process with MediaPipe
    results = pose.process(image_rgb)

    if not results.pose_landmarks:
        return False, "no_person_detected"

    # Continue to next steps...
```

Step 2: Body Visibility Check

```python
def has_full_body(landmarks):
    left_ankle = landmarks[LEFT_ANKLE]
    right_ankle = landmarks[RIGHT_ANKLE]

    # Check if ankles are visible and not cut off
    ankles_visible = (
        left_ankle.visibility > 0.5 and
        right_ankle.visibility > 0.5 and
        left_ankle.y < 0.95 and  # Not at bottom edge
        right_ankle.y < 0.95
    )

    return ankles_visible
```

Step 3: Shooting Pose Verification

```python
def is_shooting_pose(landmarks):
    left_wrist = landmarks[LEFT_WRIST]
    right_wrist = landmarks[RIGHT_WRIST]
    left_shoulder = landmarks[LEFT_SHOULDER]
    right_shoulder = landmarks[RIGHT_SHOULDER]
    nose = landmarks[NOSE]

    # Check if at least one wrist is above shoulders
    left_wrist_up = left_wrist.y < left_shoulder.y
    right_wrist_up = right_wrist.y < right_shoulder.y

    # Check if wrist is near head level (shooting position)
    left_near_head = abs(left_wrist.y - nose.y) < 0.3
    right_near_head = abs(right_wrist.y - nose.y) < 0.3

    return (left_wrist_up or right_wrist_up) and (left_near_head or right_near_head)
```

Step 4: Visibility Score

```python
def check_visibility(landmarks):
    visibility_scores = [lm.visibility for lm in landmarks]
    avg_visibility = np.mean(visibility_scores)

    if avg_visibility < 0.6:
        return False, "low_visibility"

    return True, "passed"
```

📁 Output Structure

Before Cleaning

```
training_data/
├── form_quality_classifier/
│   ├── good_form/
│   │   ├── 1.jpg
│   │   ├── 2.jpg  (soccer player - wrong sport)
│   │   ├── 3.jpg
│   │   └── ...
│   └── needs_work/
│       └── ...
├── raw_images/
└── api_downloads/
```

After Cleaning

```
training_data/
├── form_quality_classifier/
│   ├── good_form/
│   │   ├── 1.jpg  ✓ (kept)
│   │   ├── 3.jpg  ✓ (kept)
│   │   └── ...
│   └── needs_work/
│       └── ...
├── quarantine/
│   └── form_quality_classifier/
│       └── good_form/
│           └── 2__REASON_not_shooting_pose.jpg  ✗ (removed)
├── DATASET_CLEANUP_REPORT.md
└── DATASET_CLEANUP_REPORT.json
```

📋 Cleanup Report Format

Markdown Report (DATASET_CLEANUP_REPORT.md)

```markdown
# Dataset Cleanup Report
Generated: 2025-12-13 18:30:00

## Summary
- Total Images Scanned: 19,451
- Images Kept: 5,230
- Images Removed: 14,221
- Removal Rate: 73.12%

## Removal Reasons
- not_shooting_pose: 6,234 images
- no_person_detected: 3,456 images
- partial_body: 2,890 images
- low_visibility: 1,234 images
- too_small: 407 images

## By Directory
### form_quality_classifier/good_form
- Total: 320
- Kept: 89
- Removed: 231 (72.19%)
```

JSON Report (DATASET_CLEANUP_REPORT.json)

```json
{
  "cleanup_date": "2025-12-13T18:30:00",
  "summary": {
    "total_images_scanned": 19451,
    "images_kept": 5230,
    "images_removed": 14221,
    "removal_rate": "73.12%"
  },
  "removal_reasons": {
    "not_shooting_pose": 6234,
    "no_person_detected": 3456,
    "partial_body": 2890,
    "low_visibility": 1234,
    "too_small": 407
  },
  "by_directory": {
    "form_quality_classifier/good_form": {
      "total": 320,
      "kept": 89,
      "removed": 231
    }
  }
}
```

🔄 Manual Review Process

Step 1: Review Quarantine

```
cd training_data/quarantine
find . -name "*__REASON_not_shooting_pose*" | head -20
```

Step 2: Restore False Positives

```
# If image was incorrectly removed
mv quarantine/path/to/image__REASON_*.jpg original/path/to/image.jpg
```

Step 3: Verify Kept Images

```
# Sample 20 random kept images
find form_quality_classifier/good_form -type f | shuf -n 20
```

⚙️ Configuration Options

Adjust Thresholds

```
# In clean_dataset.py

# Detection confidence (0.0 - 1.0)
min_detection_confidence = 0.5  # Default
min_detection_confidence = 0.7  # Stricter

# Minimum image dimensions
MIN_WIDTH = 200   # Default
MIN_HEIGHT = 200  # Default

# Visibility threshold
MIN_AVG_VISIBILITY = 0.6  # Default
MIN_AVG_VISIBILITY = 0.7  # Stricter

# Shooting pose threshold (wrist-to-head distance)
HEAD_PROXIMITY_THRESHOLD = 0.3  # Default (30% of image height)
```

Modify Directories to Clean

```
# In main() function
directories_to_clean = [
    "form_quality_classifier/good_form",
    "form_quality_classifier/needs_work",
    "raw_images",
    "api_downloads",
    # Add or remove directories as needed
]
```

📈 Expected Results

Dataset Quality Improvements

- **Consistency:** All images show individual shooters
- **Completeness:** Full body visible in all images
- **Relevance:** Only basketball shooting poses
- **Quality:** High visibility and adequate resolution

Training Benefits

- **Faster Convergence:** Clean data = faster training
- **Better Accuracy:** Model learns correct patterns
- **Fewer Errors:** No confusion from wrong sports
- **Improved Generalization:** Consistent shooting poses

🚨 Important Notes

Safety Features

1. **No Permanent Deletion:** All removed images moved to quarantine
2. **Detailed Logging:** Every removal decision is logged
3. **Reason Codes:** Each removed file tagged with reason
4. **Reversible:** Manual review can restore false positives

Performance Considerations

- **Processing Time:** ~2-3 seconds per image
- **Total Time (19,451 images):** ~10-15 hours
- **Recommend:** Run overnight or in batches
- **Progress Updates:** Every 100 images

Disk Space

- **Quarantine Size:** ~70-75% of original dataset
- **Recommended:** Ensure 2x dataset size free space
- **Cleanup:** After manual review, delete quarantine folder

🔍 Quality Assurance Checklist

After cleaning, verify:

- [ ] Report shows expected removal rate (70-75%)
- [ ] Quarantine folder contains removed images
- [ ] Sample 50 kept images manually - all valid?
- [ ] Sample 50 quarantined images - correctly removed?
- [ ] Check removal reasons distribution - reasonable?
- [ ] Verify no corrupted files in kept dataset
- [ ] Document any systematic issues found

- [ ] Restore any false positives from quarantine

📚 Related Documentation

- `SKELETON_OVERLAY_IMPLEMENTATION.md` - Pose detection system
- `DATASET_PREPARATION_GUIDE.md` - Original dataset collection
- `training_data/README.md` - Dataset overview

🎓 Best Practices

Before Running Cleanup

1. **Backup:** Create backup of training_data/
2. **Disk Space:** Verify sufficient space available
3. **Test Run:** Test on small subset first
4. **Time:** Plan for long processing time

During Cleanup

1. **Monitor:** Check progress logs periodically
2. **Stop Safely:** Use Ctrl+C if needed (safe to interrupt)
3. **Resources:** Monitor CPU/RAM usage

After Cleanup

1. **Review Report:** Analyze removal statistics
2. **Manual Sample:** Verify random samples
3. **Restore Errors:** Fix any false positives
4. **Update Docs:** Document any findings

🐛 Troubleshooting

Issue: MediaPipe installation errors

```
pip install --upgrade mediapipe opencv-python numpy
```

Issue: Memory errors with large images

```python
# Add image resizing before processing
max_dimension = 1920
if max(height, width) > max_dimension:
    scale = max_dimension / max(height, width)
    image = cv2.resize(image, None, fx=scale, fy=scale)
```

Issue: Too many images removed

```
# Reduce detection confidence
min_detection_confidence = 0.3  # More lenient
MIN_AVG_VISIBILITY = 0.4        # Lower threshold
```

Issue: Processing too slow

```
# Reduce model complexity
mp_pose.Pose(
    model_complexity=1,  # Faster but less accurate
    # ... other settings
)
```

📊 Validation Metrics

After cleaning, calculate:

```
# Dataset statistics
total_kept = len(find_all_images(training_data))
removal_rate = (total_removed / total_scanned) * 100

# Quality metrics (manual sample of 100 images)
true_positives = images_correctly_kept
false_positives = bad_images_kept
false_negatives = good_images_removed

precision = true_positives / (true_positives + false_positives)
recall = true_positives / (true_positives + false_negatives)
```

📝 Version History

v1.0.0 (2025-12-13)

- ✅ Initial implementation
- ✅ MediaPipe-based filtering
- ✅ Shooting pose detection
- ✅ Full body visibility checks
- ✅ Quarantine system
- ✅ Detailed reporting

👥 Support

For issues or questions:
1. Check troubleshooting section
2. Review sample images in quarantine

3. Adjust thresholds in configuration
4. Document systematic problems found

📄 Summary

The dataset cleaning tool provides:
- **Automated filtering** using MediaPipe Pose
- **Safe quarantine** system (no permanent deletion)
- **Detailed reporting** with reason codes
- **Manual review** capability for false positives
- **Quality assurance** through sampling

Result: Clean, consistent dataset of individual basketball shooters with full body visible and clear shooting motion.