Press Release

Title:

Bundesliga AI Hackathon – Challenge 1: Player Pose Classification for Media Day Archive

Overview:

Our solution automates the tedious manual process of sorting thousands of football players' images by leveraging state-of-the-art transfer learning and data augmentation techniques. By re-purposing a pre-trained ResNet50 network with custom classifier heads, we achieve robust classification performance on a sparse dataset featuring 408 images across 17 pose-based classes. Our approach not only delivers high overall accuracy but also provides detailed per-class precision, recall, and F1 scores, ensuring that the model generalizes well to unseen images.

FAQ

Q1: What is the main problem being addressed?

The challenge targets the inefficiency of manually sorting Media Day archive images. Our system automates the classification of player poses—such as full body, head shot, and various arm configurations—thus streamlining the archival process.

Q2: What technical approach has been implemented?

- **Transfer Learning:** We leveraged a a pre-trained ResNet50 model and fine-tuned it by freezing most layers (except for layer4 and the final fully connected layers). A custom classification heads was added to adapt the model to our specific task.
- **Stratified Fold:** Given the limited dataset size, we used stratified splitting to ensure that training, validation, and test sets preserve the original class distribution, reducing the risk of bias during model evaluation.
- **Early Stopping:** To prevent overfitting, we implemented early stopping.
- **Regularization (Dropout & Batch Normalization):** To improve generalization, we applied Dropout layers in the custom classifier head, randomly deactivating neurons during training. Additionally, Batch Normalization was incorporated to stabilize and accelerate the learning process by normalizing layer inputs.
- **Data Augmentation:** Techniques including random rotations, color jitter, and random affine transformations are applied to mitigate data sparsity.
- **Hyperparameter Tuning:** We utilized Optuna for automated hyperparameter optimization. Key parameters like learning rate, batch size, dropout rate, and early stopping patience were tuned to achieve optimal performance.
- **Synthetic Data Generation Attempt:**
    - *StyleGAN-ADA:* We experimented with StyleGAN-ADA to generate synthetic images and enrich our dataset, given its limited size. Initial results were promising; however, the training process was hindered by memory limitations in Jupyter Lab's file explorer due to frequent snapshot saves of the network's state. This experience highlighted the need to redesign our training workflow to avoid such bottlenecks in the future. The current model status is attached.

- *Stable Diffusion Models*: Additionally, we explored using stable diffusion models for synthetic image generation. Unfortunately, all the new stable diffusion models we tried were too memory intensive.

Q3: What are the key evaluation metrics?

Overall Accuracy:

Our model achieved an overall accuracy of **98%**, demonstrating strong generalization across the test set.

Per-Class Metrics:

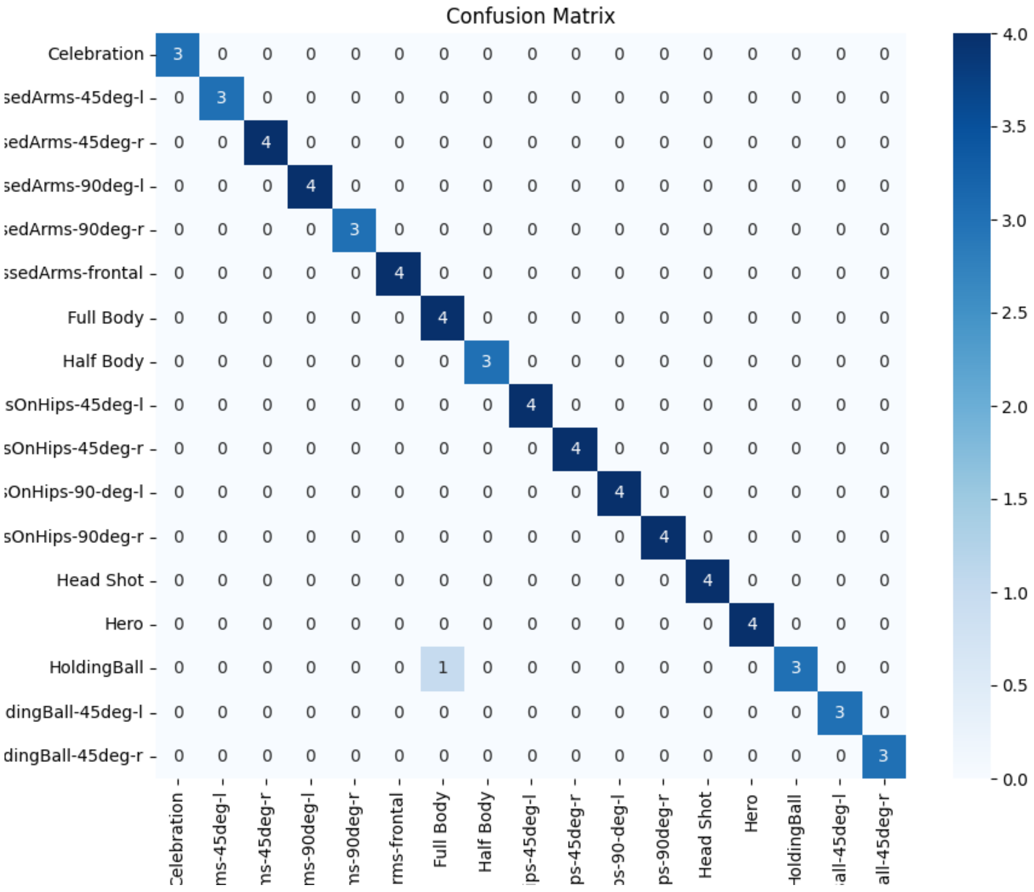
For each of the 17 classes, we calculated precision, recall and F1-score.

- **Celebration: Precision:** 1.00, Recall: 1.00, F1-score: 1.00 (Support: 3)
- **CrossedArms-45deg-l:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 3)
- **CrossedArms-45deg-r:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 4)
- **CrossedArms-90deg-l:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 4)
- **CrossedArms-90deg-r**: Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 3)
- **CrossedArms-frontal:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 4)
- **Full Body:** Precision: 0.80, Recall: 1.00, F1-score: 0.89 (Support: 4)
- **Half Body:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 3)
- **HandsOnHips-45deg-l:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 4)
- **HandsOnHips-45deg-r:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 4)
- **HandsOnHips-90-deg-l:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 4)
- **HandsOnHips-90deg-r:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 4)
- **Head Shot:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 4)
- **Hero:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 4)
- **HoldingBall:** Precision: 1.00, Recall: 0.75, F1-score: 0.86 (Support: 4)
- **HoldingBall-45deg-l:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 3)
- **HoldingBall-45deg-r:** Precision: 1.00, Recall: 1.00, F1-score: 1.00 (Support: 3)

In summary, the macro average scores are approximately 0.99 for precision, recall, and F1-score, while the weighted average scores are around 0.99 for precision, 0.98 for recall, and 0.98 for F1-score.

Confusion Matrix

We also generated a confusion matrix to visualize misclassifications and better understand any edge cases. The attached image clearly illustrates the performance across all classes.



	Celebration	ms-45deg-l	ns-45deg-r	ms-90deg-l	ns-90deg-r	rms-frontal	Full Body	Half Body	ips-45deg-l	ps-45deg-r	ps-90-deg-l	ps-90deg-r	Head Shot	Hero	HoldingBall	all-45deg-l	all-45deg-r
Celebration	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sedArms-45deg-l	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sedArms-45deg-r	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sedArms-90deg-l	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0
sedArms-90deg-r	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0
ssedArms-frontal	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0
Full Body	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0
Half Body	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
sOnHips-45deg-l	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0
sOnHips-45deg-r	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0
sOnHips-90-deg-l	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0
sOnHips-90deg-r	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
Head Shot	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0
Hero	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0
HoldingBall	0	0	0	0	0	0	1	0	0	0	0	0	0	0	3	0	0
dingBall-45deg-l	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0
dingBall-45deg-r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3

Q4: What were the main challenges?

- **Data Sparsity:** With only 24 images per class, it was crucial to find various methods to prevent overfitting.
- **Synthetic Data Generation:** We explored methods to increase dataset diversity using synthetic image generation. Initially, we employed a conditional GAN (StyleGAN-ADA), which showed promising results, but the training was interrupted by memory limitations—stemming from frequent snapshot saves in Jupyter Lab's file explorer. In parallel, we also experimented with stable diffusion models; however, the latest models proved too memory intensive for our current infrastructure, prompting us to try older versions. Ultimately, due to time and budget constraints, we were unable to fully train these models. Nevertheless, the insights gained will guide us in redesigning our training process in future experiments to better address these challenges.

Q5: How can others reproduce your results?

We have ensured that our solution is fully reproducible:

For further details have a look at the **Reproduction Document.pdf**

Q6: What were approaches we pursued but did not work?

- **Synthetic Data Generation:** In our efforts to increase dataset diversity, we experimented with synthetic image generation using both a conditional GAN (StyleGAN-ADA) and stable diffusion models. Our initial work with StyleGAN-ADA showed promising results; however, the training process was interrupted due to memory limitations—primarily caused by the regular snapshot saves in Jupyter Lab's file explorer. In parallel, we explored stable diffusion models, but these proved too memory intensive for our current setup.