RecogniChess

An unsupervised domain-adaptation approach to chessboard recognition

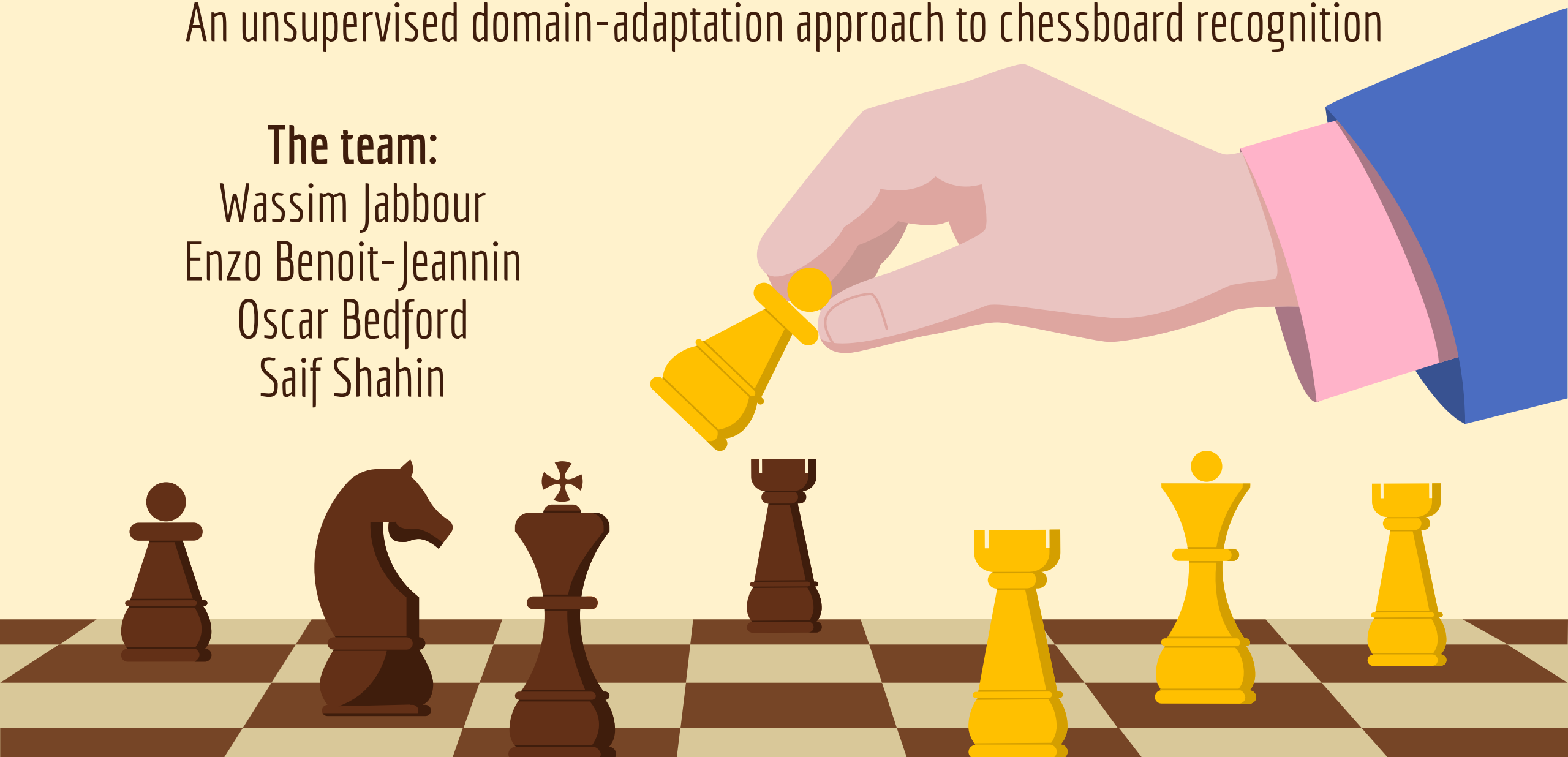
The team:

Wassim Jabbour

Enzo Benoit-Jeannin

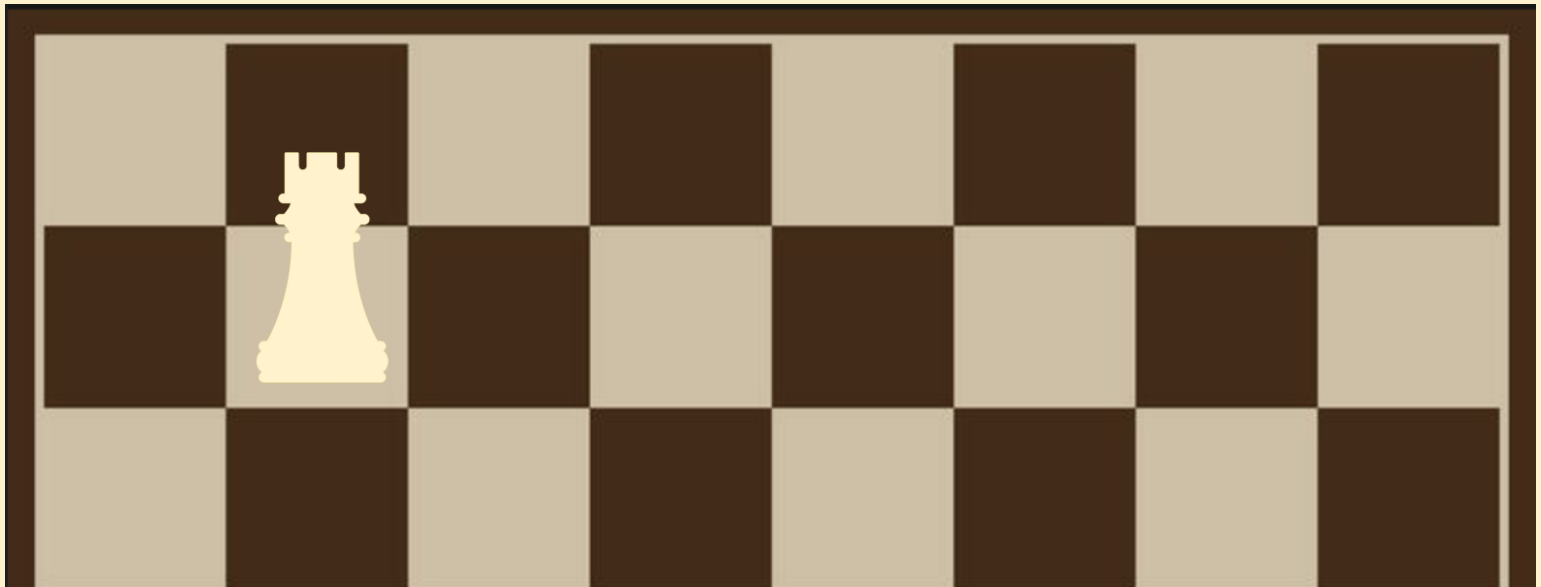
Oscar Bedford

Saif Shahin



Introduction

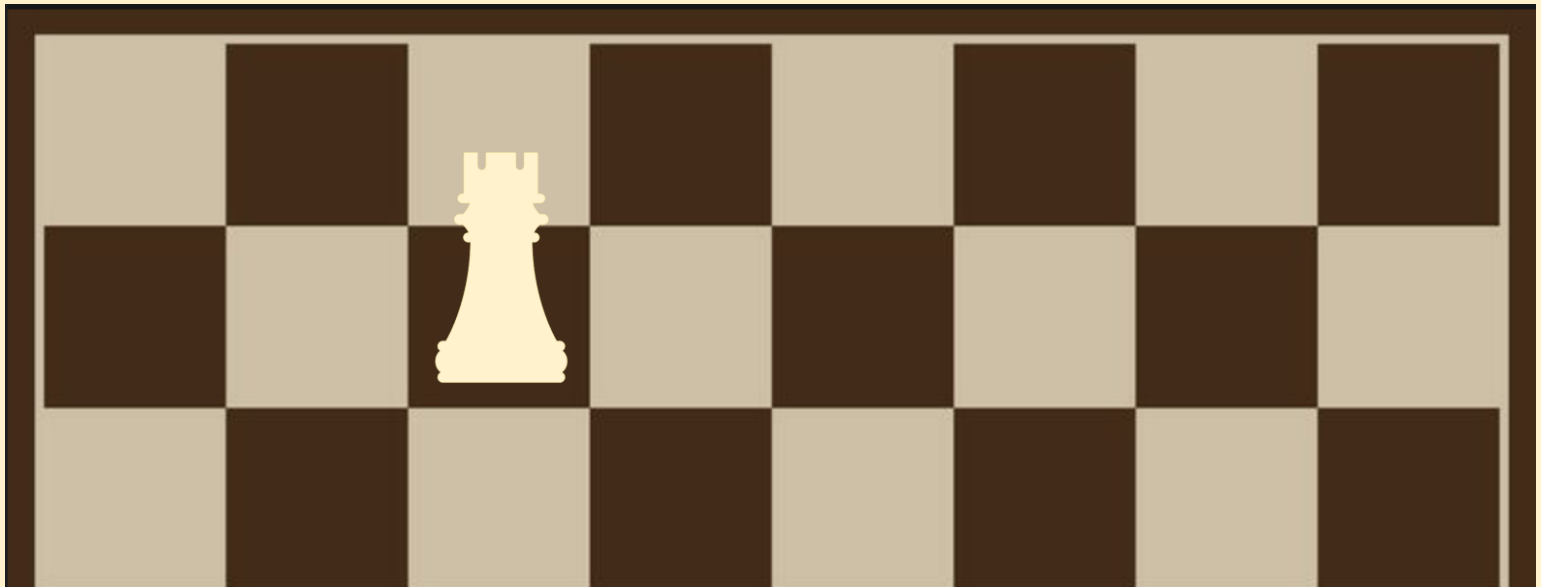
Chess move annotation is important



Introduction

Chess move annotation is important

Automation can be highly beneficial

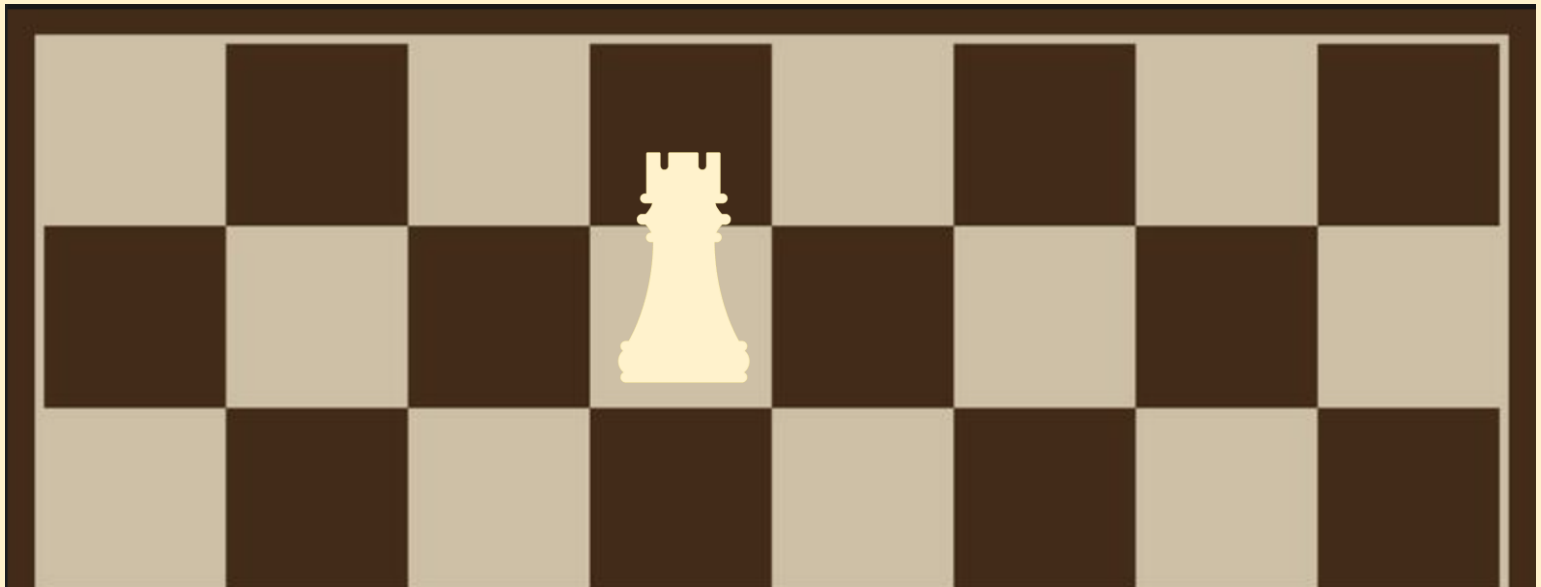


Introduction

Chess move annotation is important

Automation can be highly beneficial

Requires a lot of labeled data



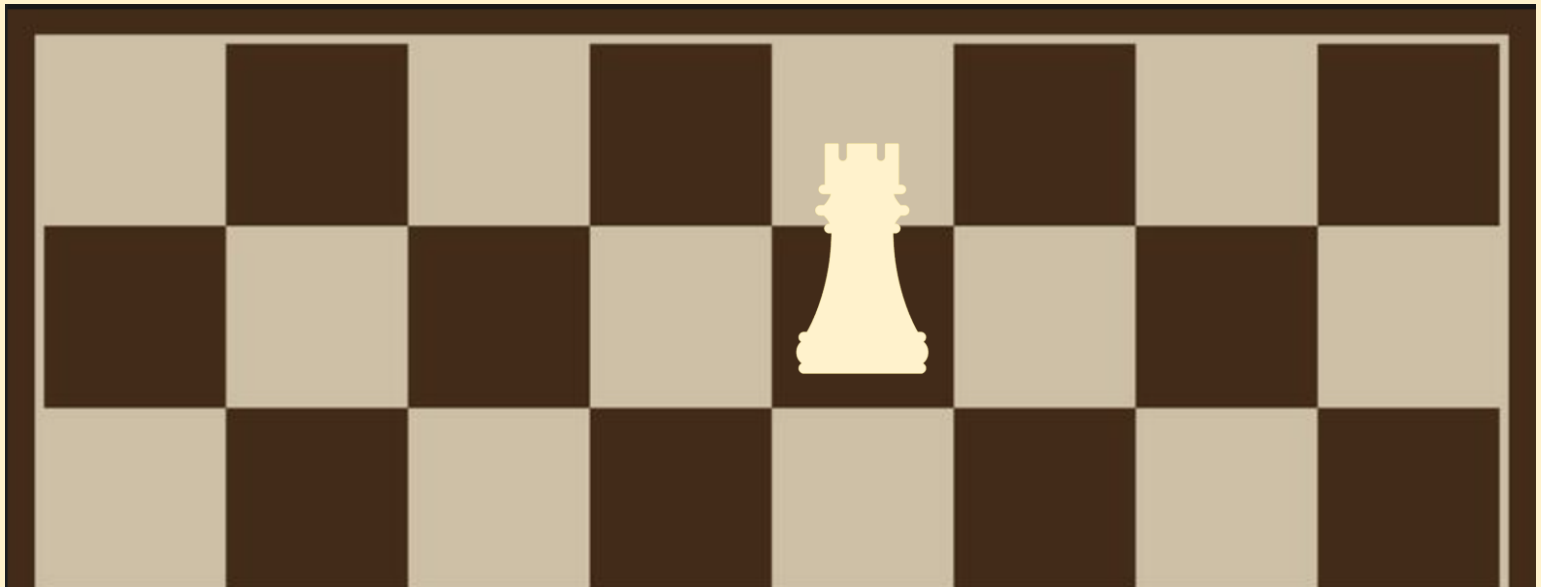
Introduction

Chess move annotation is important

Automation can be highly beneficial

Requires a lot of labeled data

Labeled data is hard to come by



Introduction

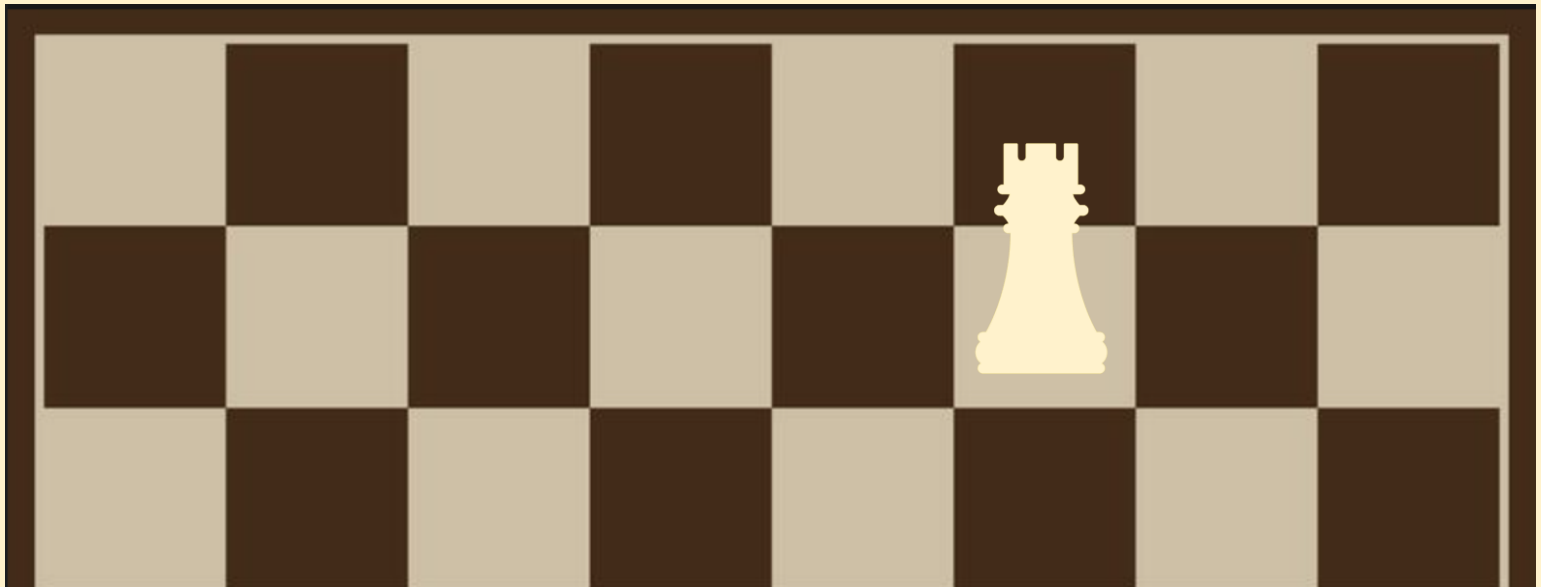
Chess move annotation is important

Automation can be highly beneficial

Requires a lot of labeled data

Labeled data is hard to come by

Labelling chess board images is tedious



Introduction

Chess move annotation is important

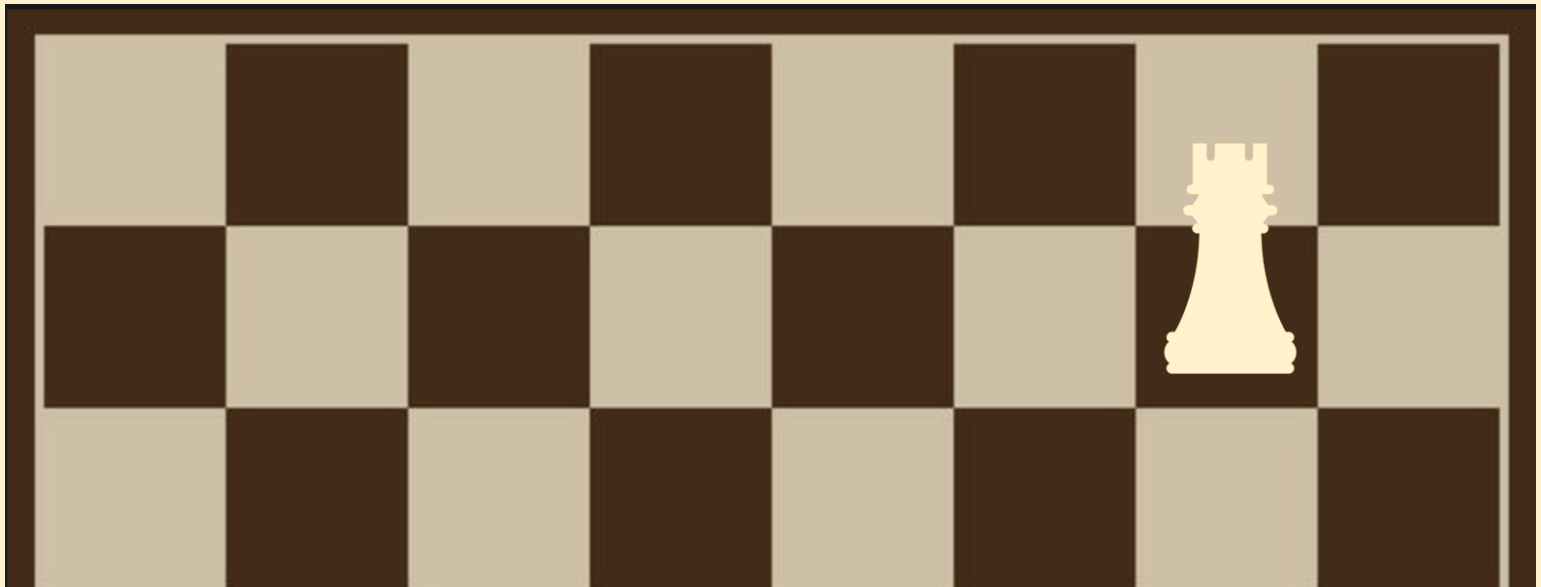
Automation can be highly beneficial

Requires a lot of labeled data

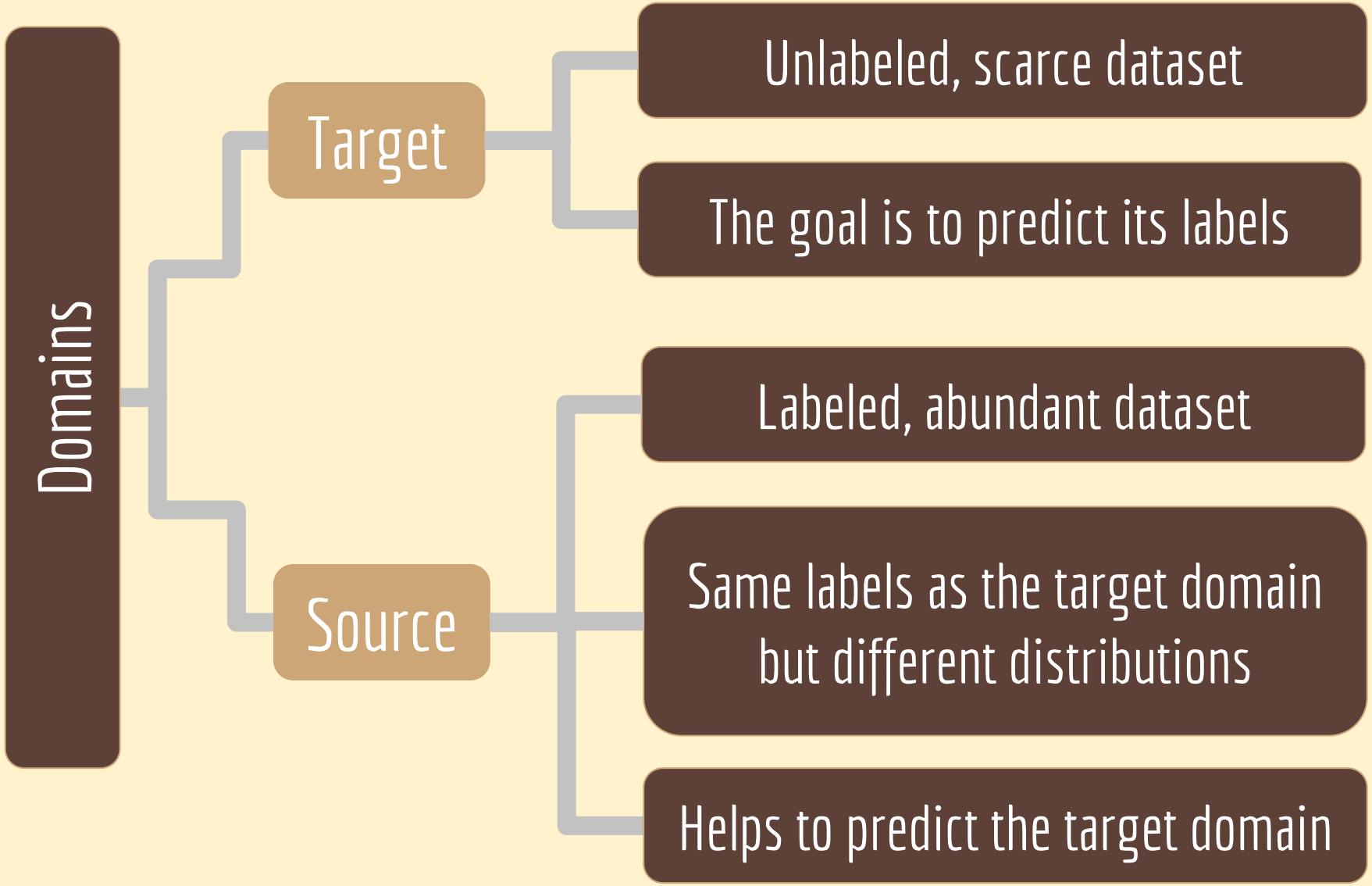
Labeled data is hard to come by

Labelling chess board images is tedious

Unsupervised domain adaptation



Overview of unsupervised DA[1]



[1] S. Ben-David et al., "A theory of learning from different domains," Mach. Learn., vol. 79, pp. 151-175, 2010.

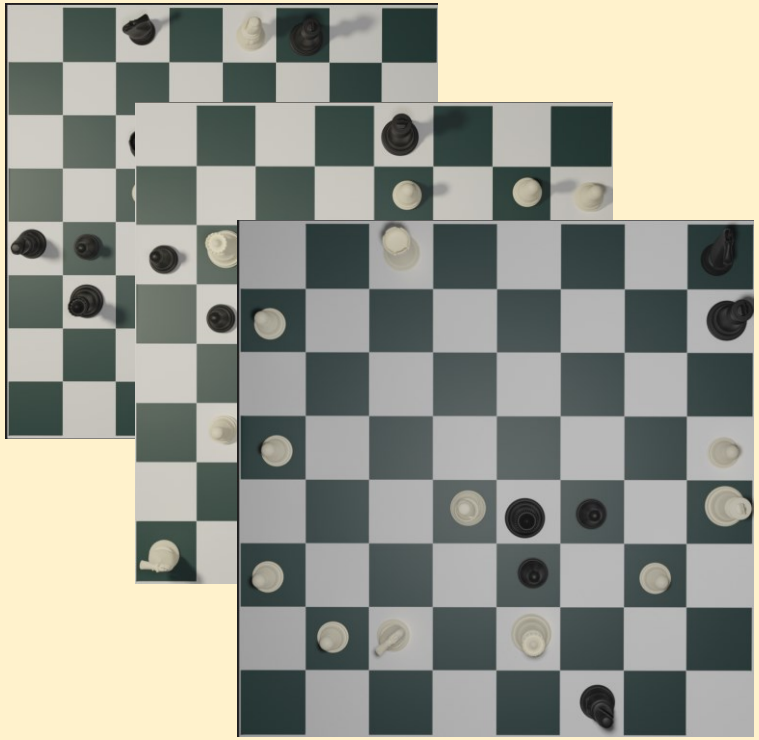
Unsupervised DA in the context of chess

Capture unprocessed, unlabeled, top-view chessboard images

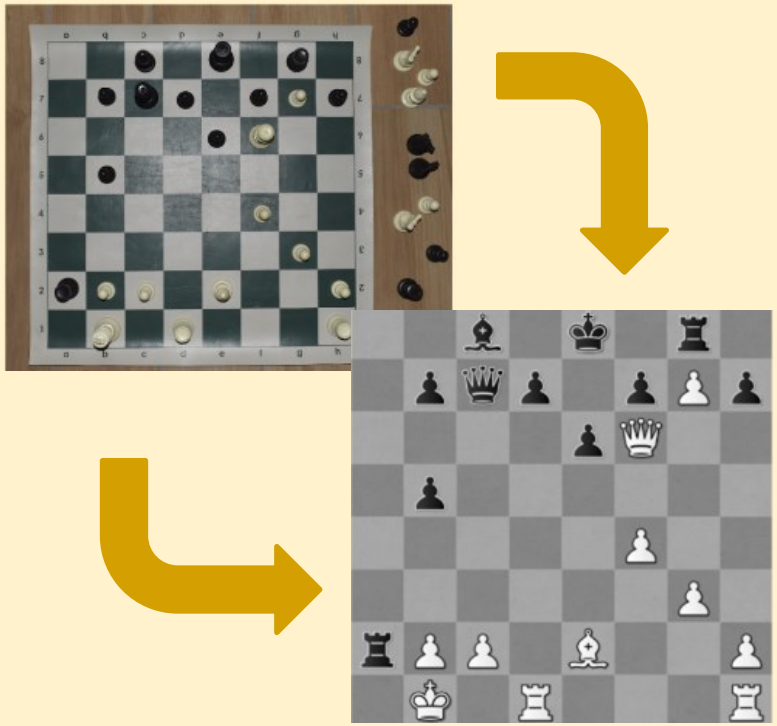
Generate labeled images with similar feature / target distributions

Build a pipeline that utilizes the source domain inputs to predict the target domain

Target domain



Source domain



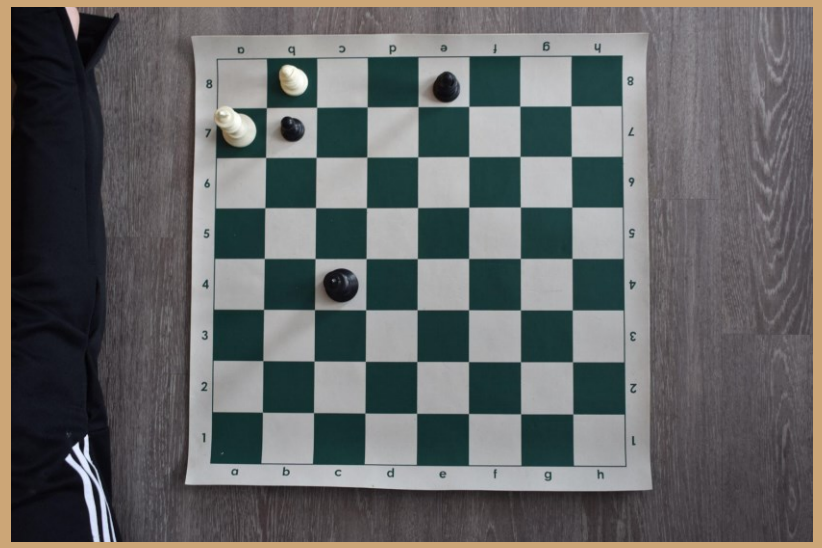
Using Unsupervised DA

Target domain dataset

500 full, distinct, legal chess positions

Unlabelled (for the most part)

Different lighting and background conditions



[2] A. D. S. D. Neto and R. M. Campello, "Chess position identification using pieces classification based on synthetic images generation and deep neural network fine-tuning," in Proc. 21st Symp. Virtual Augmented Reality (SVR), pp. 152-160, IEEE, Oct. 2019.

Source domain dataset

Generated using Blender

4500 distinct, labelled, and legal chess positions pre-processed into 288,000 individual squares

Different lighting conditions

Different centering and rotation of the pieces

Multiple top-view camera angles

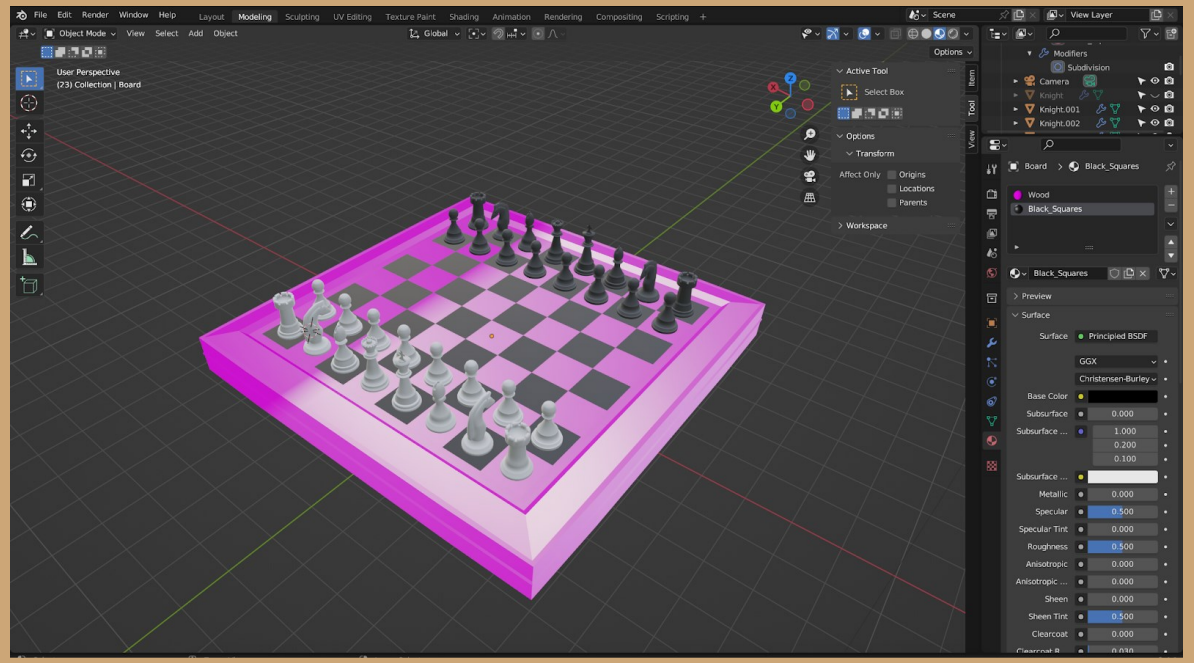
Makes the trained model invariant to all of the above conditions

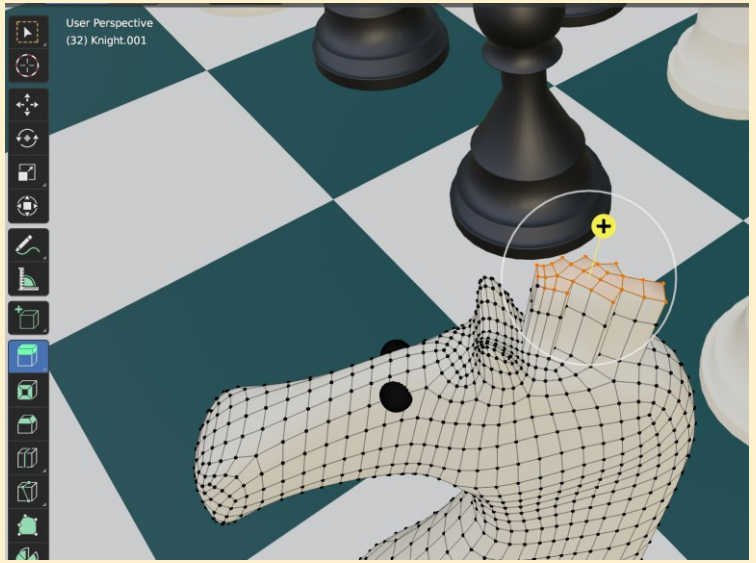
Data generation details

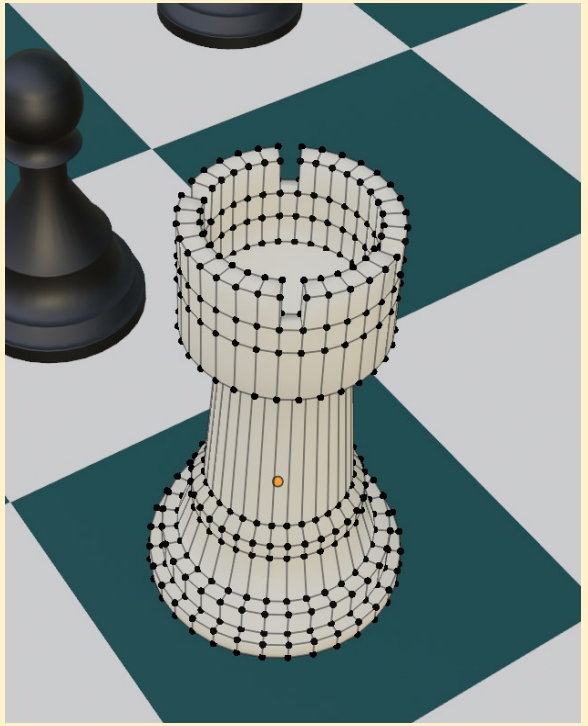
Utilized Blender to generate top-view chessboard images

Tuned the texture and shapes of the pieces to match the target dataset to the best possible extent

Camera angle and lighting were varied between positions

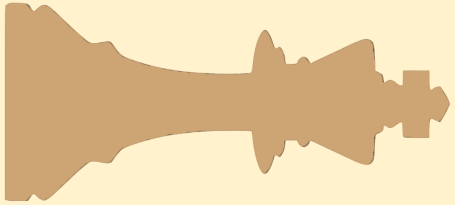






Note about generating full images & cropping them

For the synthetic dataset, full positions were generated & cropped into squares

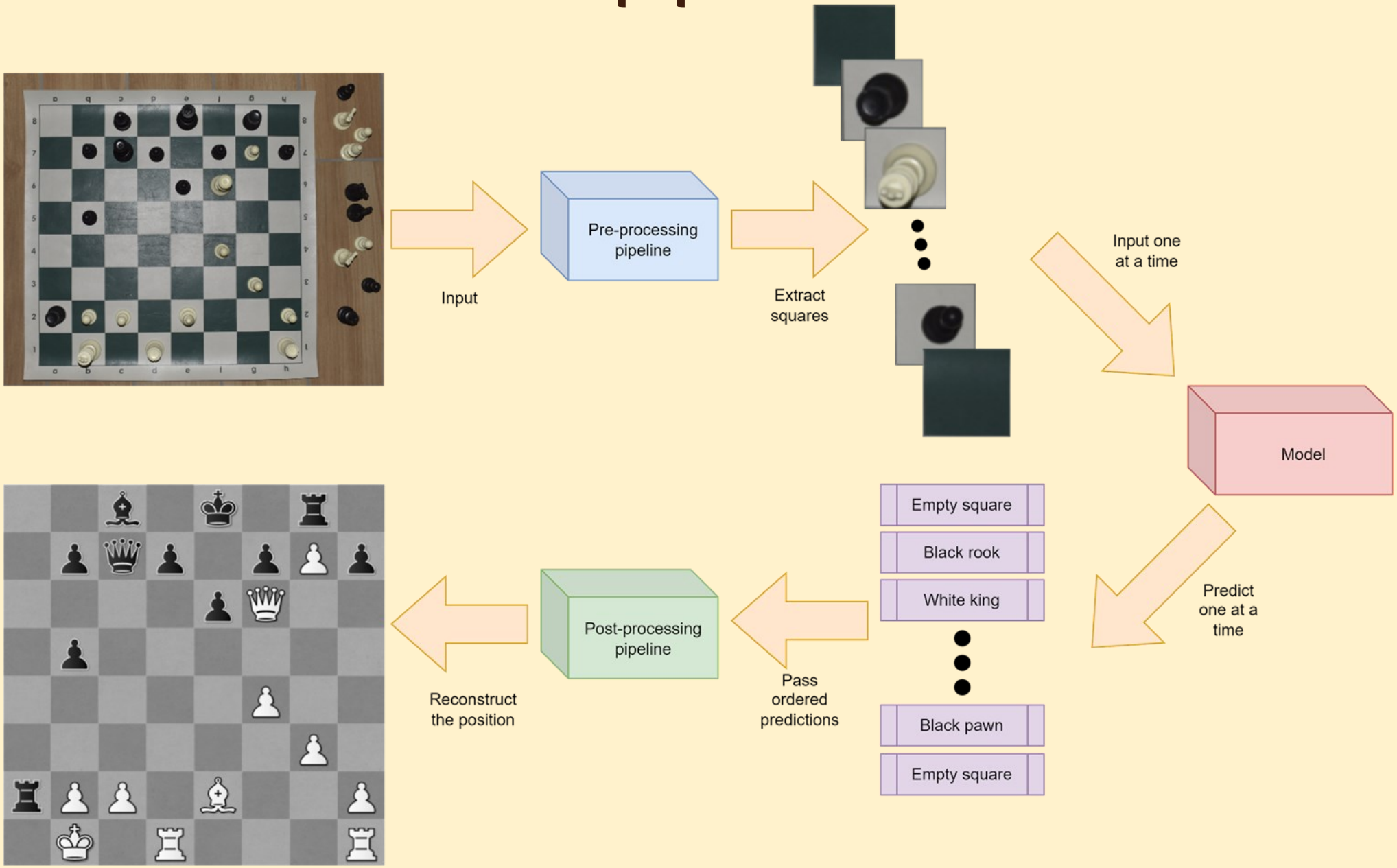


Why not generate individual squares directly ?

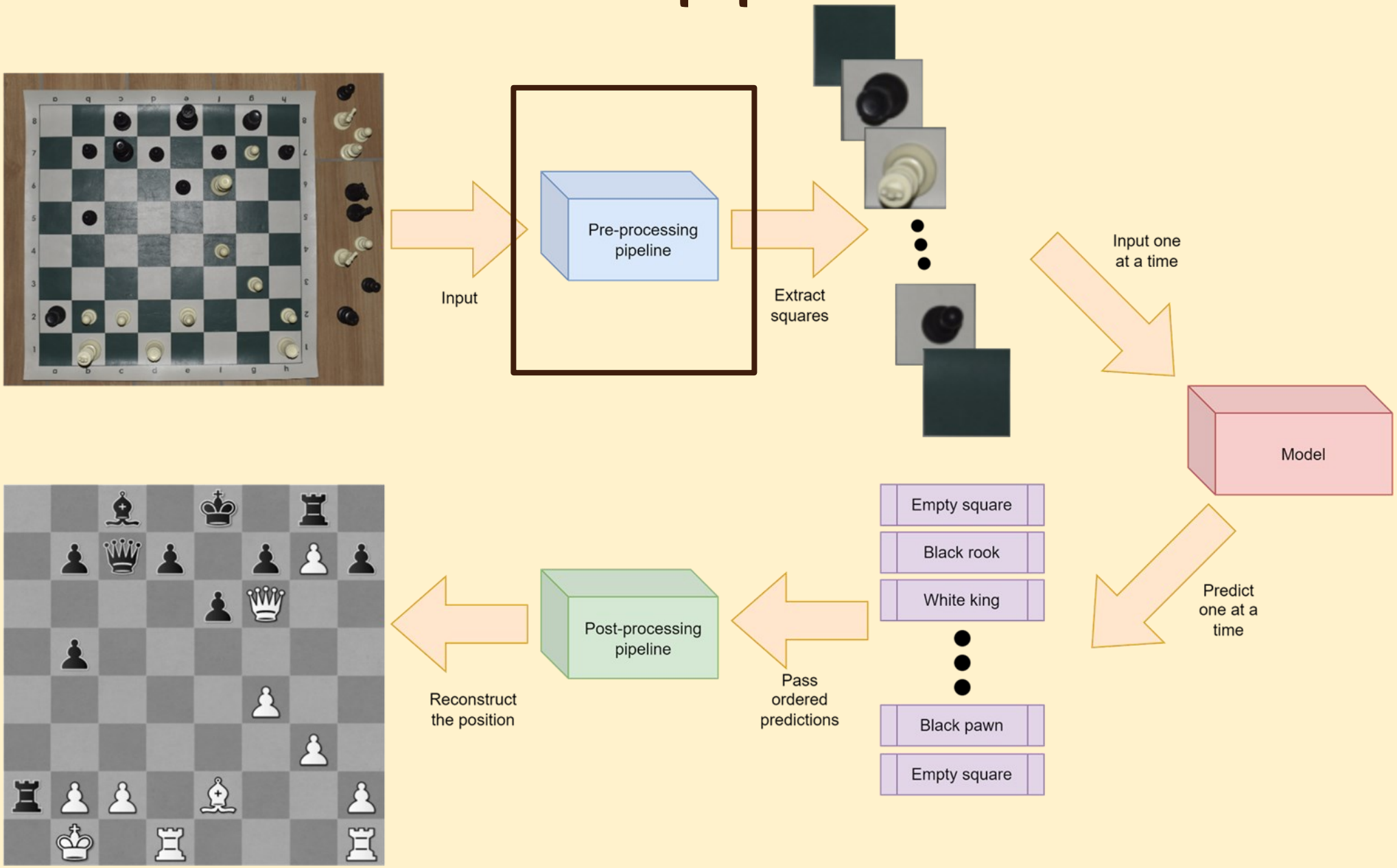
1- Generating full positions leads to more realistic lighting condition distributions, since a square must now be part of a bigger picture which is the full board.

2- All Generated positions are legal. In that sense, subsequent iterations of this work could incorporate constraints stemming from chess rules.

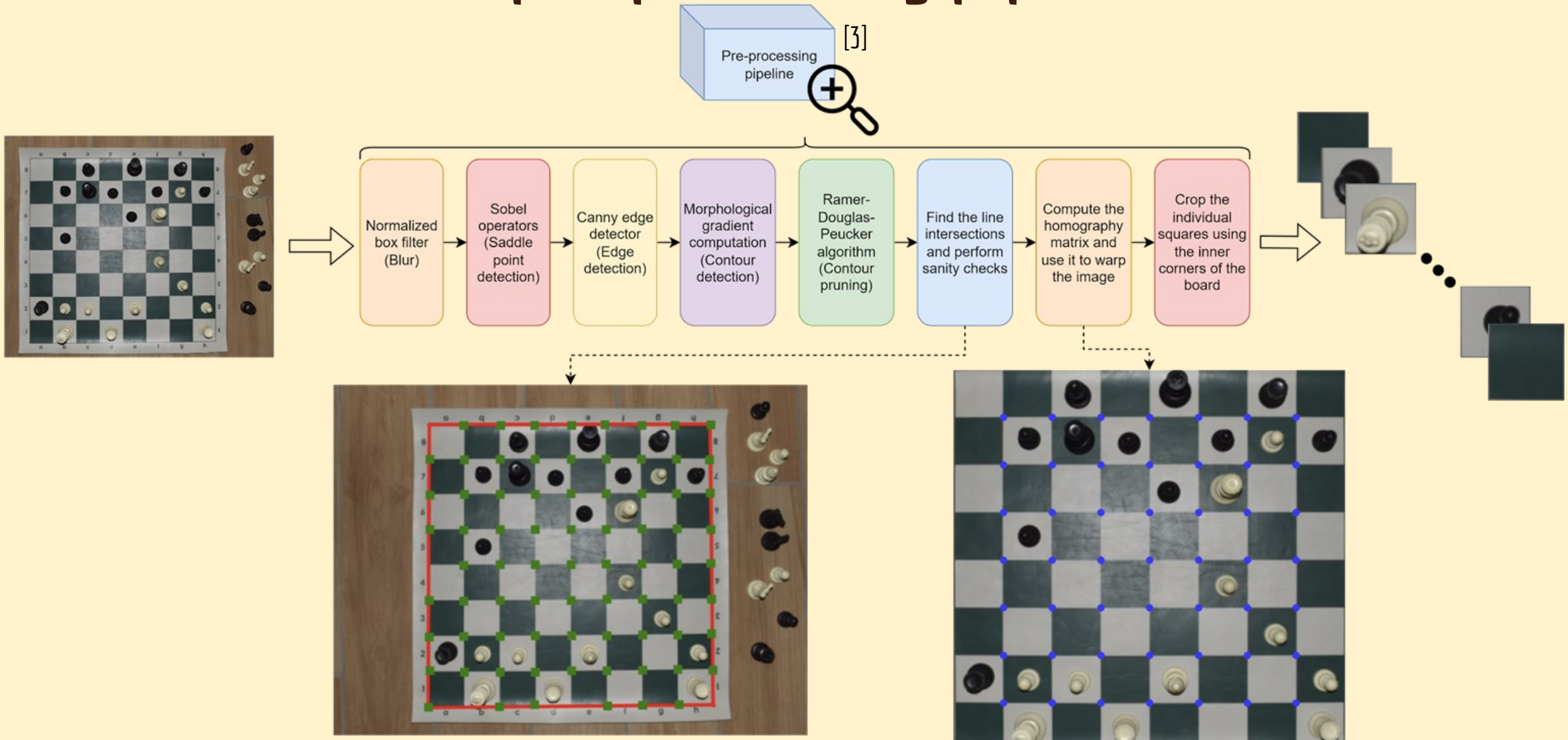
End-to-end pipeline overview



Full end-to-end pipeline overview



The pre-processing pipeline



[3]: Elucidation, "Elucidation/chessboarddetect: Hodgepodge of chessboard detection algorithms on images from actual matches.," *GitHub*. [Online]. Available: https://github.com/Elucidation/ChessboardDetect. [Accessed: 10-Apr-2023].

Interlude: Dynamic VS Static pre-processing

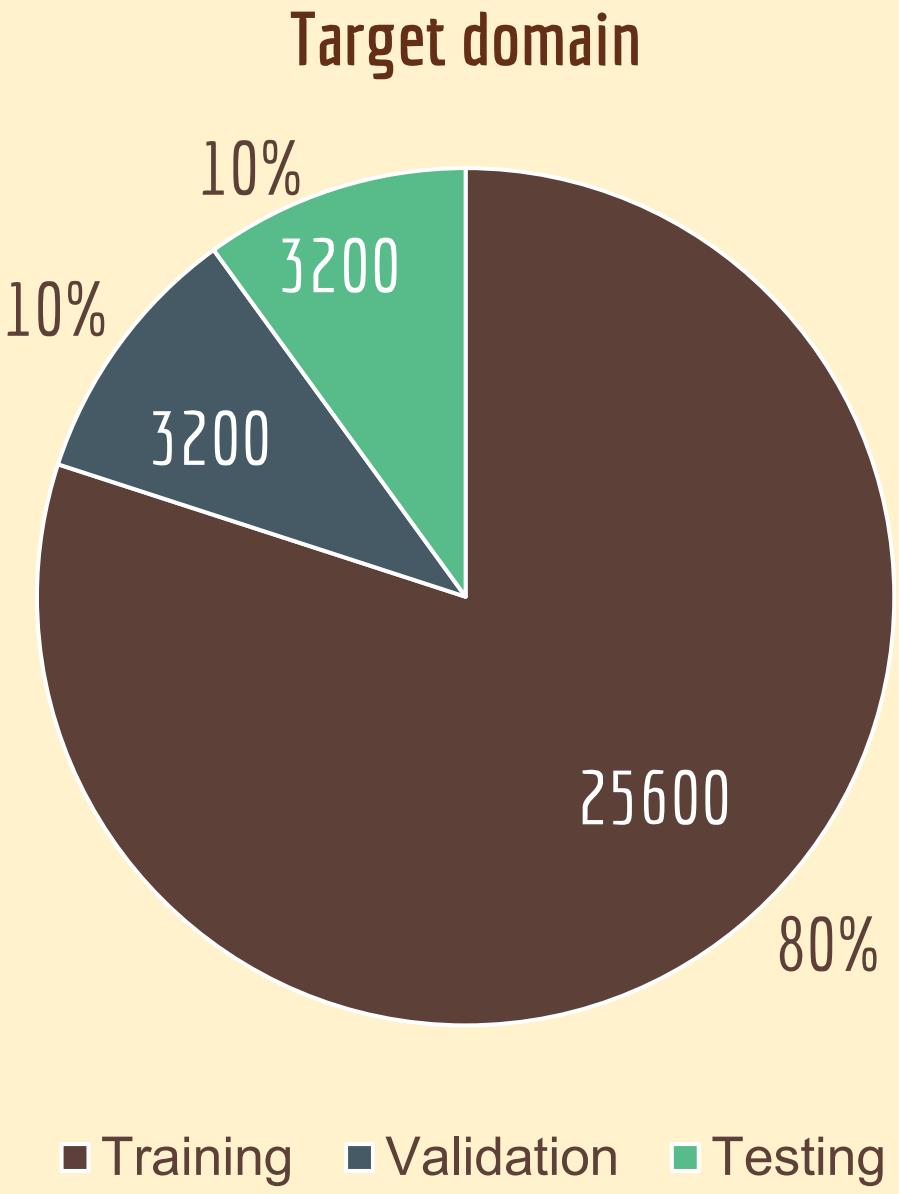
Time required to pre-process 1 image: A few seconds

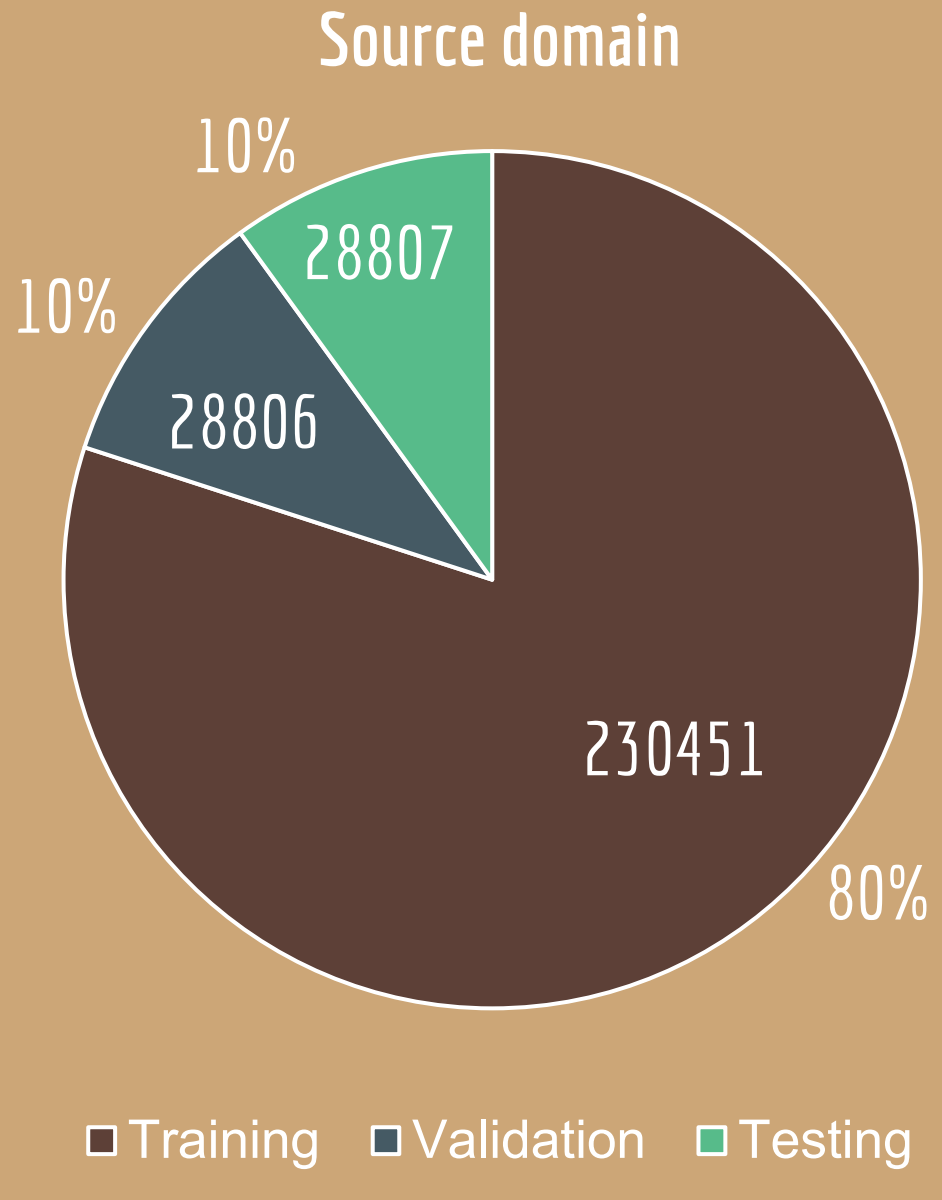
This can slow down the training drastically

In a production scenario, the pre-processing pipeline is utilized to dynamically break down an image into squares

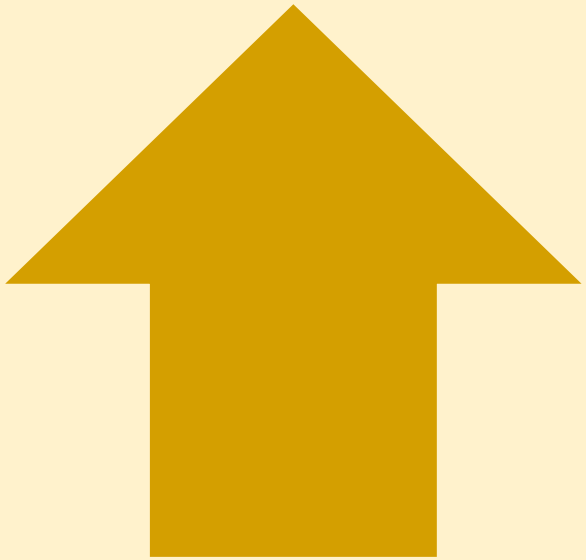
To speed up training, we statically utilized the pipeline to pre-process all the real-life data into individual squares

Interlude: Dataset splits

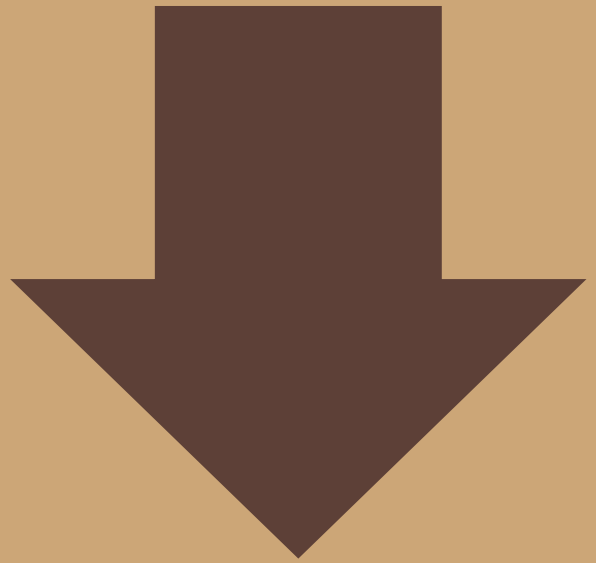




Interlude: Note about target domain labelling



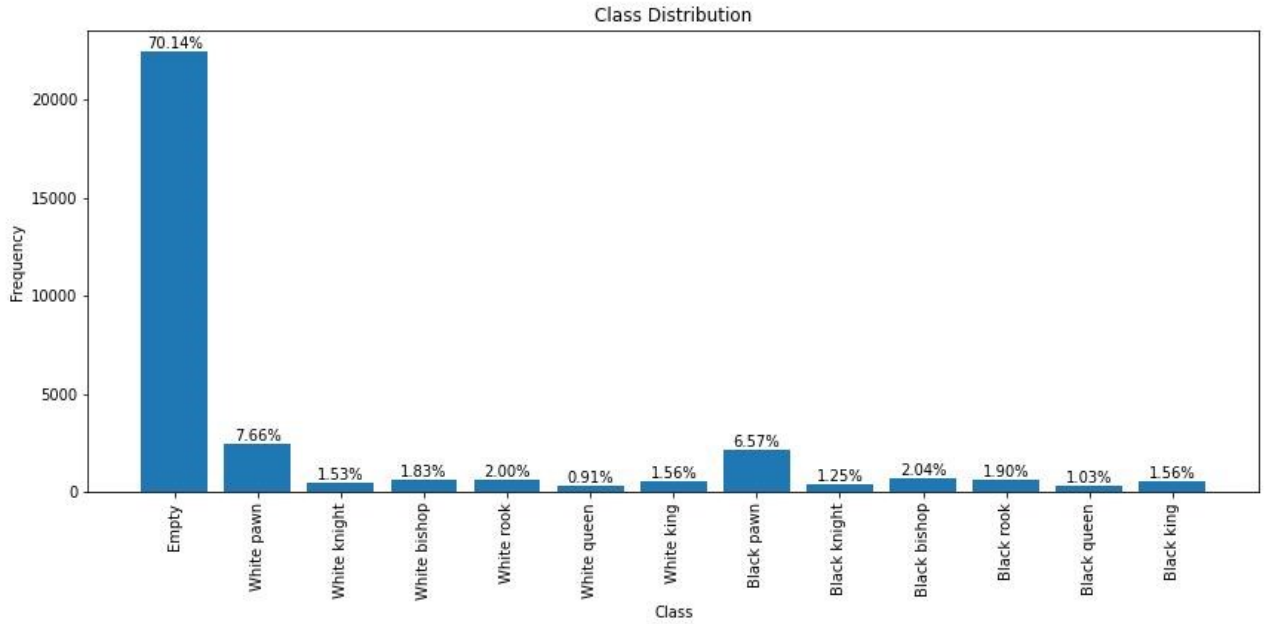
The training set of the target domain was completely unlabelled throughout the experiment



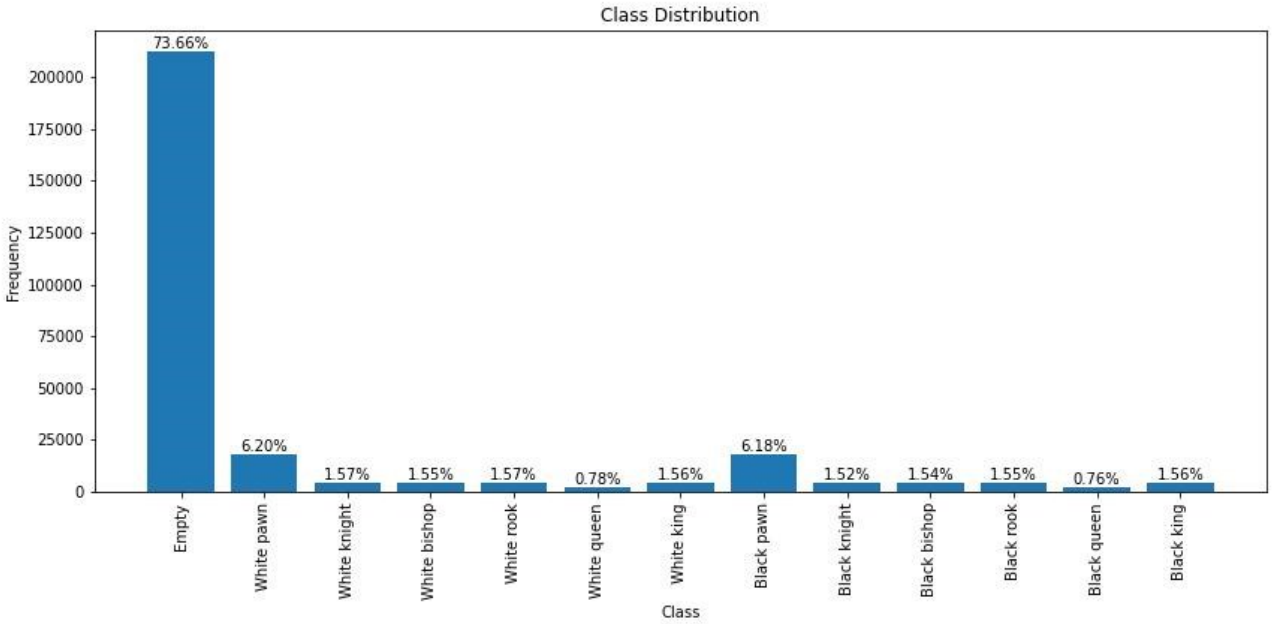
We used the validation set's labels to perform hyperparameter tuning, and for the testing set to plot metrics such as weighted F1 score & confusion matrices

Interlude: Dataset distributions

Target domain



Source domain

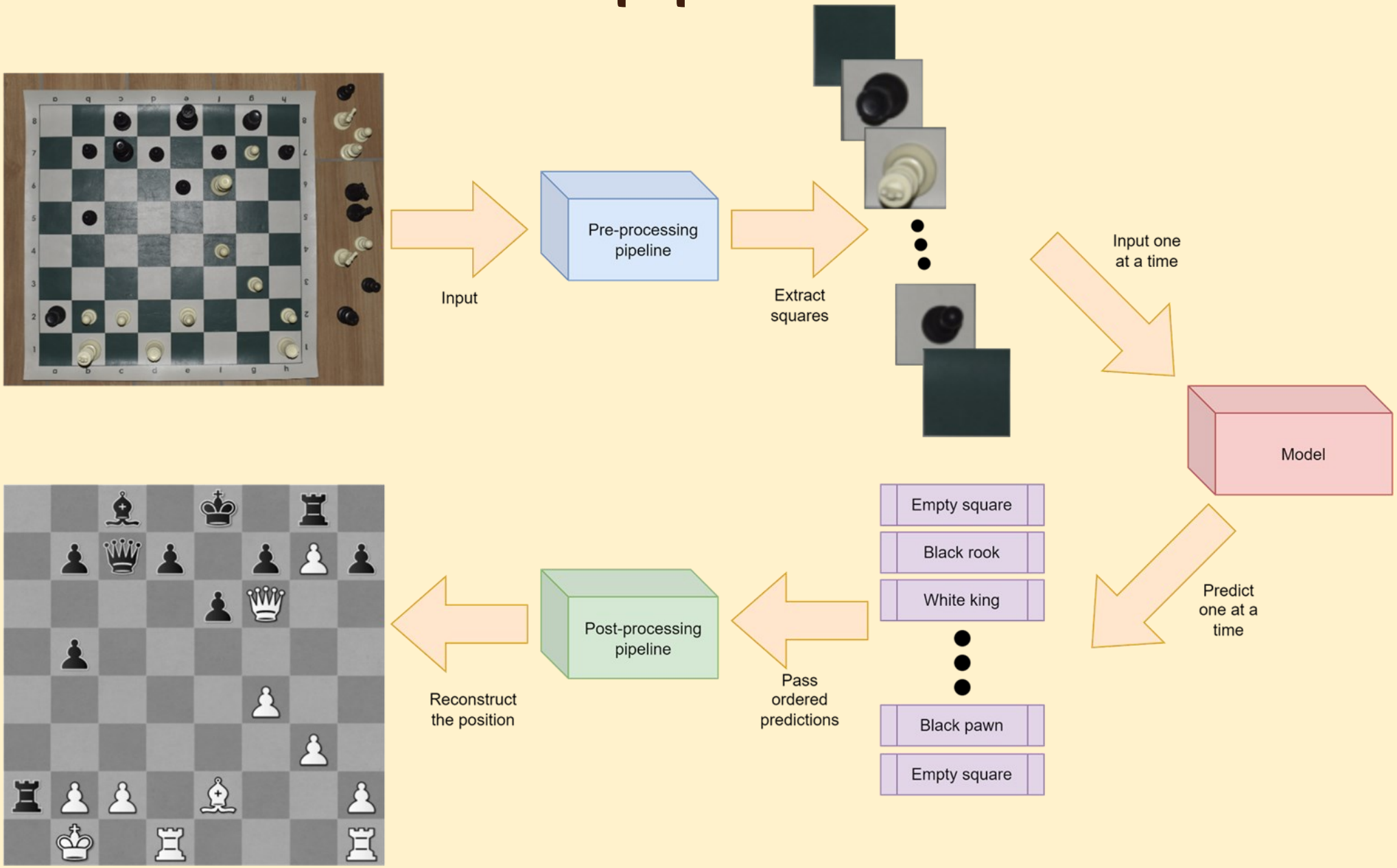


Empty squares dominate

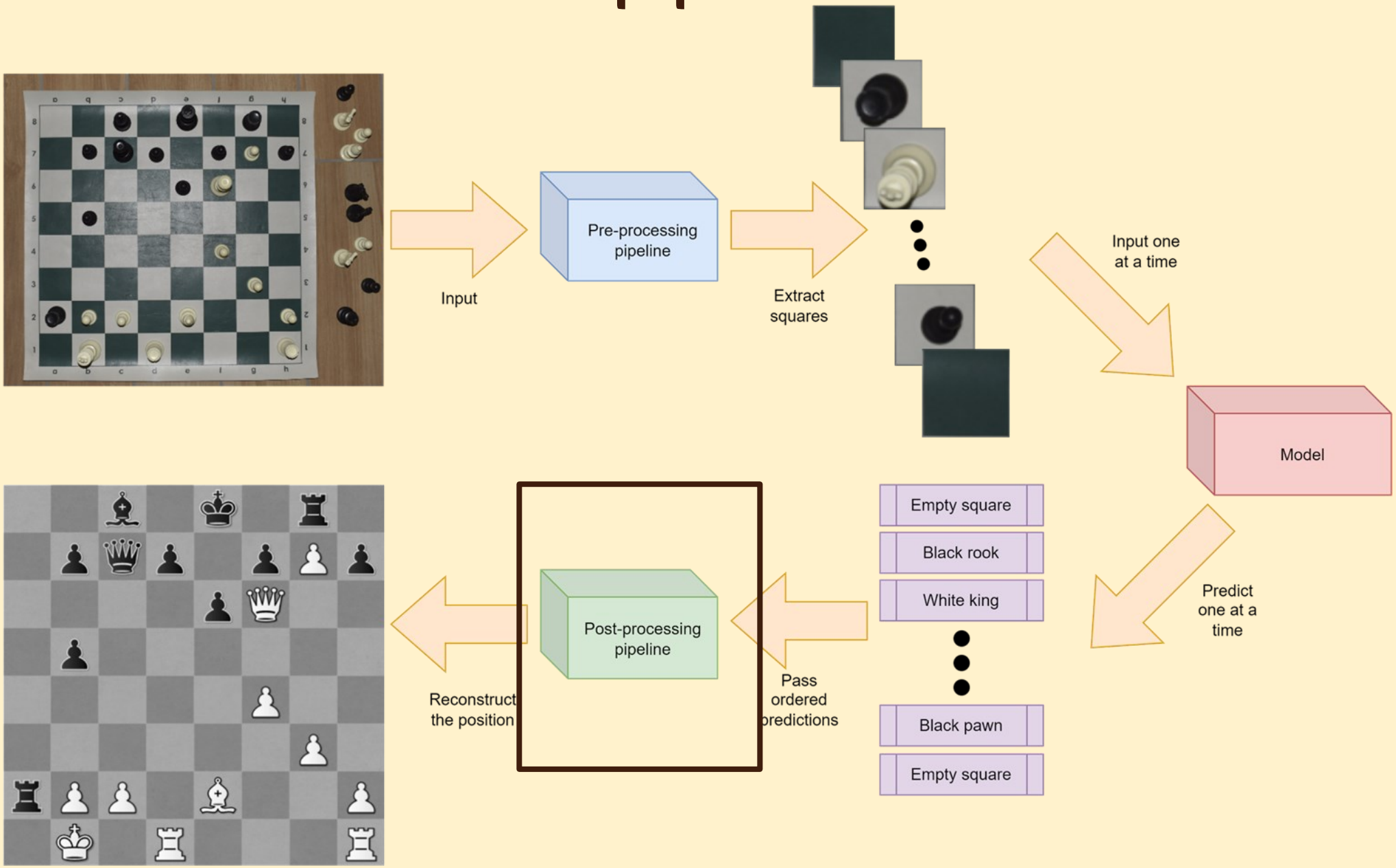
Utilized PyTorch's WeightedRandomSampler with replacement to re-balance the minibatch proportions

Augmented the data (Random flips and brightness jitters) to avoid just repeating minority class samples

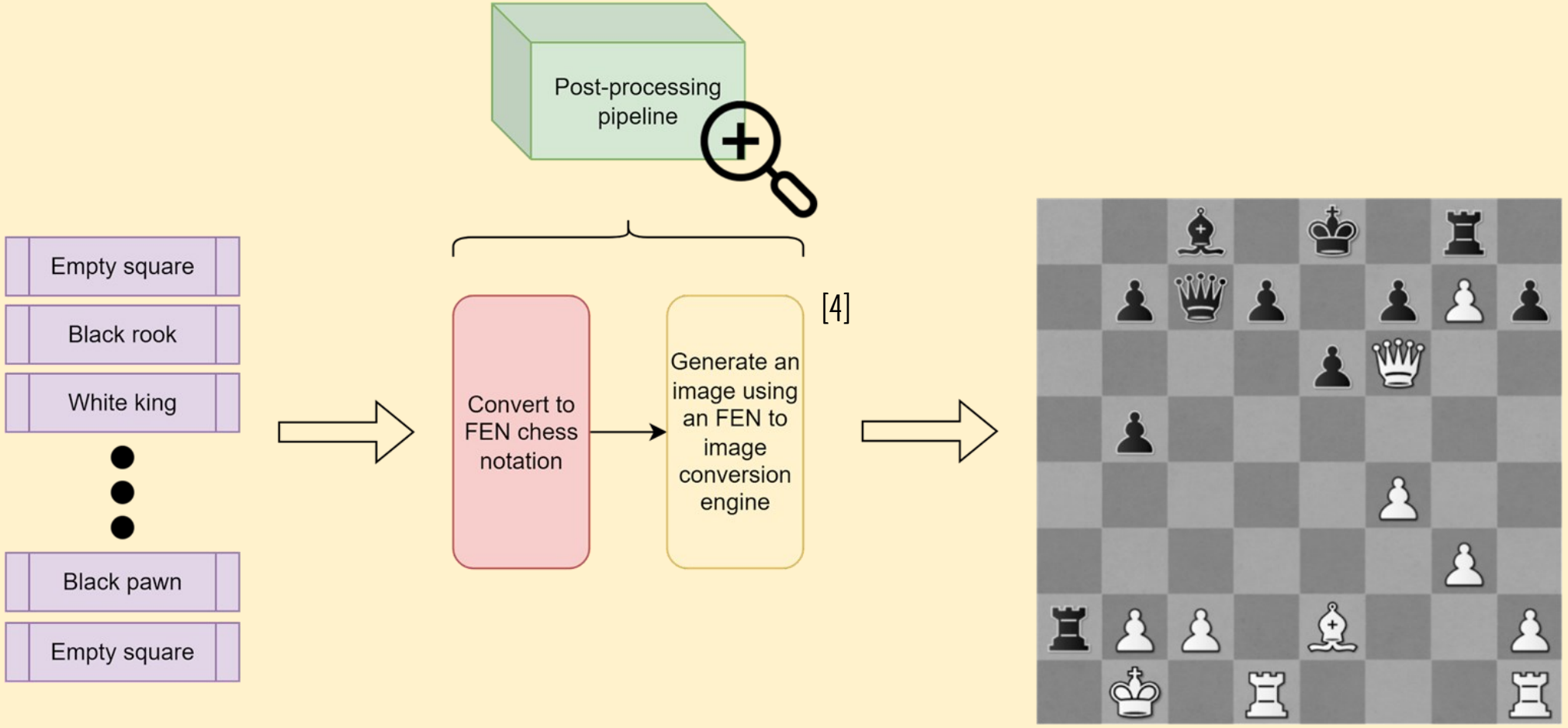
End-to-end pipeline overview



End-to-end pipeline overview



The post-processing pipeline



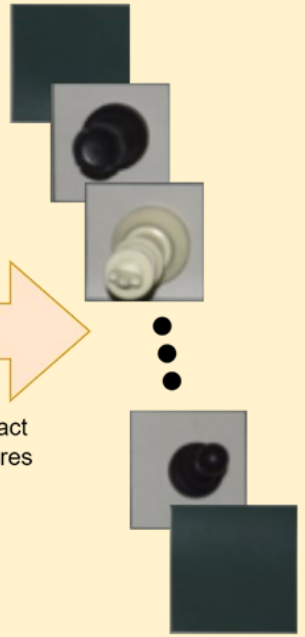
[4] “Fen to image,” Chessvisionai RSS, https://chessvision.ai/docs/tools/fen2image/ (accessed Jun. 1, 2023).

End-to-end pipeline overview

Input

Pre-processing pipeline

Extract squares



Input one at a time

Model

Predict one at a time

Empty square

Black rook

White king

Black pawn

Empty square

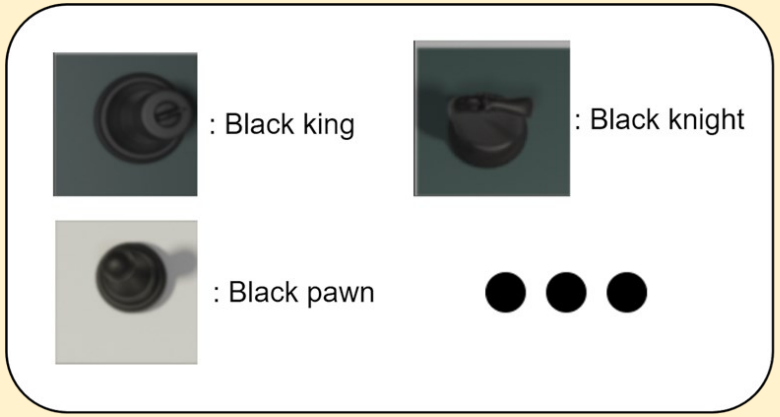
Pass ordered predictions

Post-processing pipeline

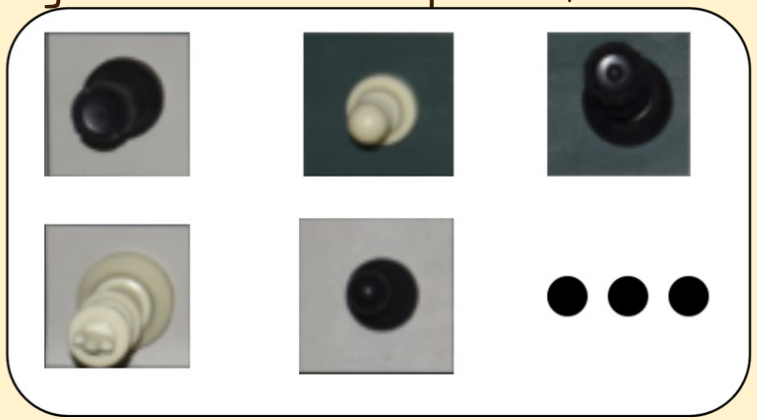
Reconstruct the position

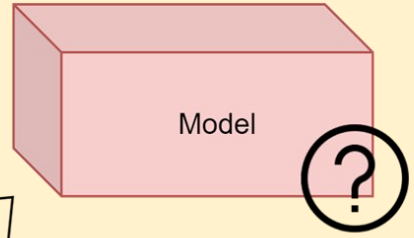
The model: Architectures

Source domain: Generated data, labelled



Target domain: Real life data, unlabelled



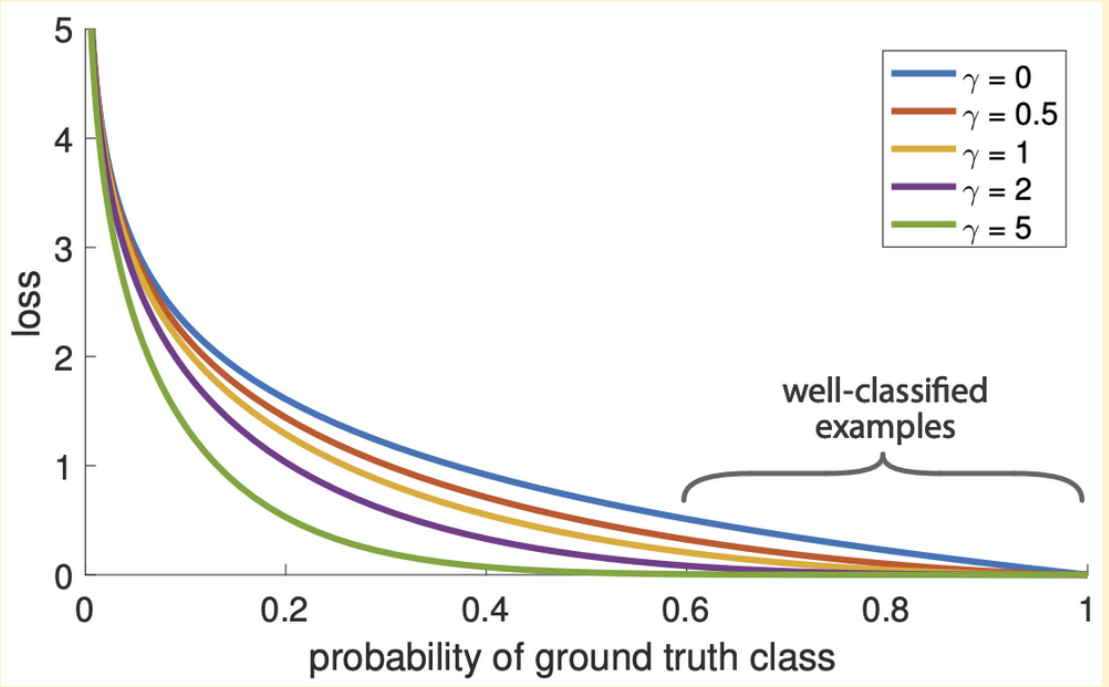


What domain adaptation technique to use in order to get good accuracies on the target domain?

Interlude: Loss functions

Cross Entropy = $-\log(p_i)$

Focal Loss[5] = $-(1-p_i)^{\gamma}\log(p_i)$



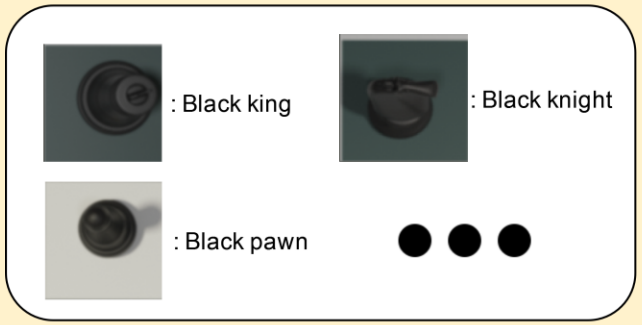
CORAL Loss[6] = $\left\|C_s - C_t\right\|_F^2$ with C(X) = $\frac{1}{n-1}(X-\overline{X})^T(X-\overline{X})$

[5] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, 'Focal Loss for Dense Object Detection', *arXiv [cs.CV]*. 2018.

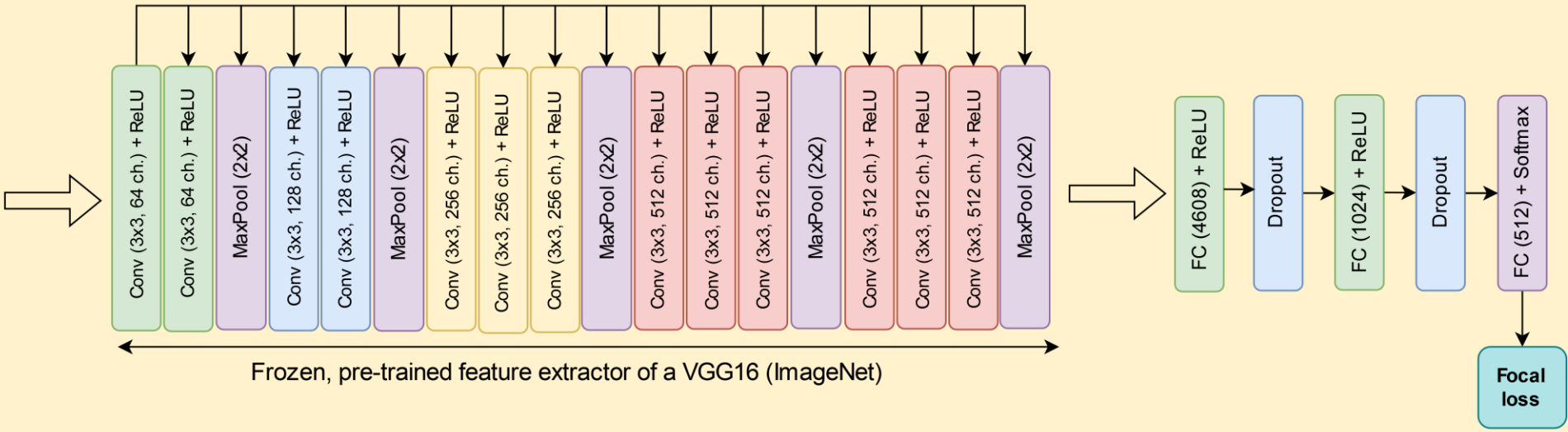
[6] B. Sun, J. Feng, and K. Saenko, 'Correlation Alignment for Unsupervised Domain Adaptation', *CoRR*, vol. abs/1612.01939, 2016.

Architecture 1: VGG16 without DA (BASE-source)

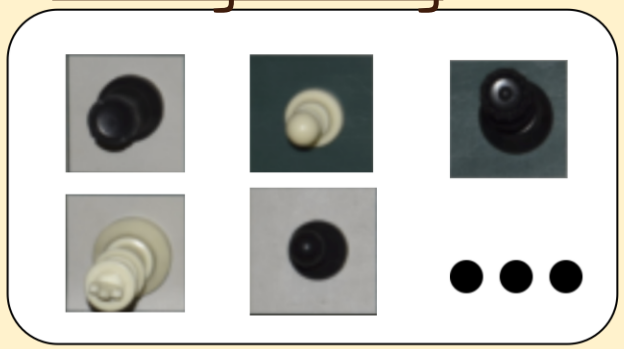
During training



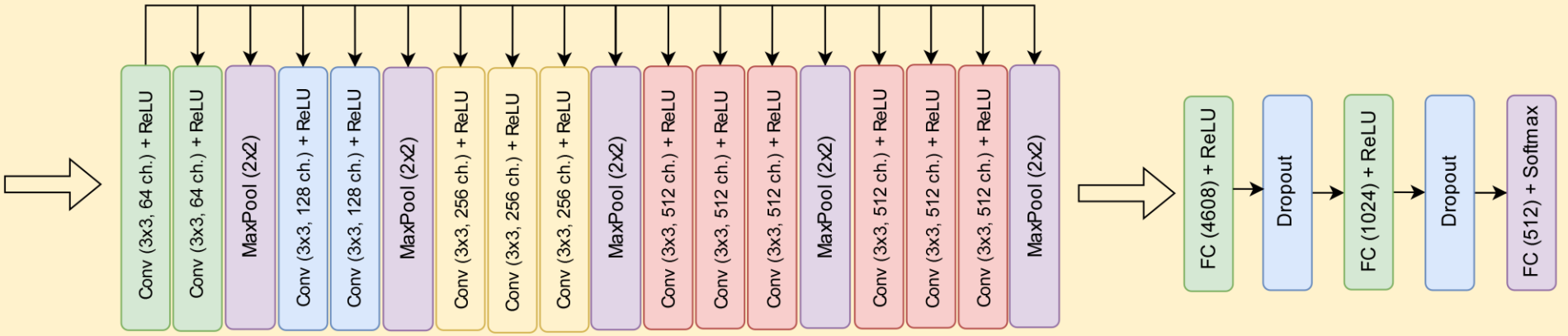
Source domain



During testing



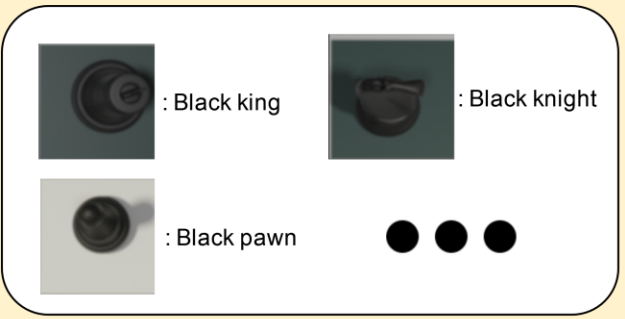
Target domain



[7] K. Simonyan and A. Zisserman, 'Very Deep Convolutional Networks for Large-Scale Image Recognition', CoRR, vol. abs/1409.1556, 2014.

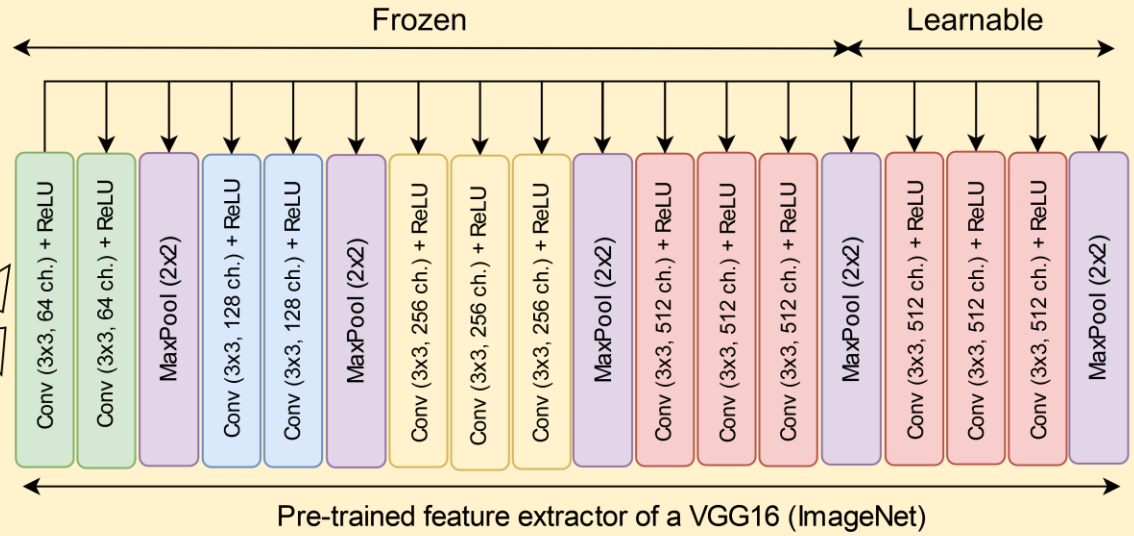
Architecture 2: Correlation alignment (CORAL)

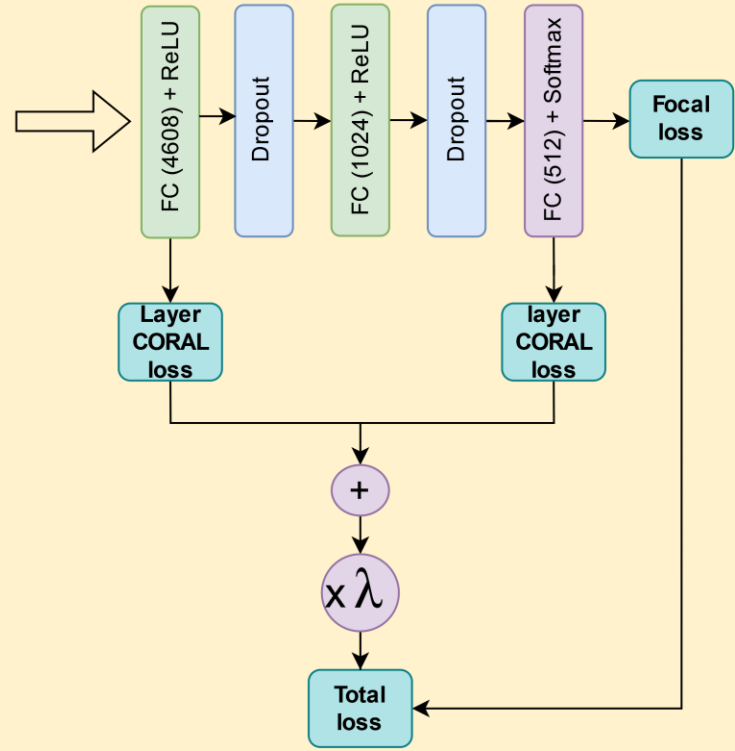
During training



Source domain

Target domain





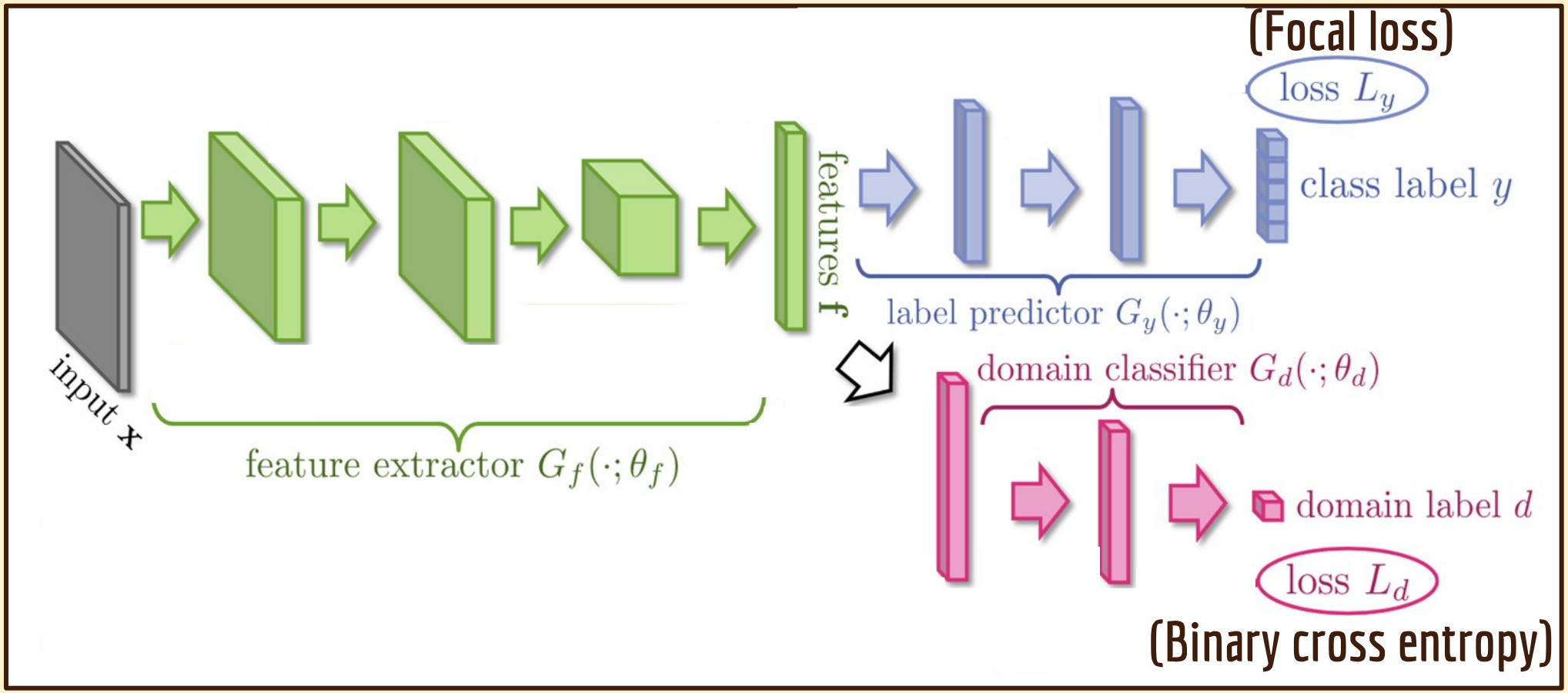
Total loss = Focal loss + λ (Coral loss)

During testing: Same as approach 1 (Target data only)

[8] B. Sun, J. Feng, and K. Saenko, 'Correlation Alignment for Unsupervised Domain Adaptation', *CoRR*, vol. abs/1612.01939, 2016.

Architecture 3: Adversarial approach (DANN)

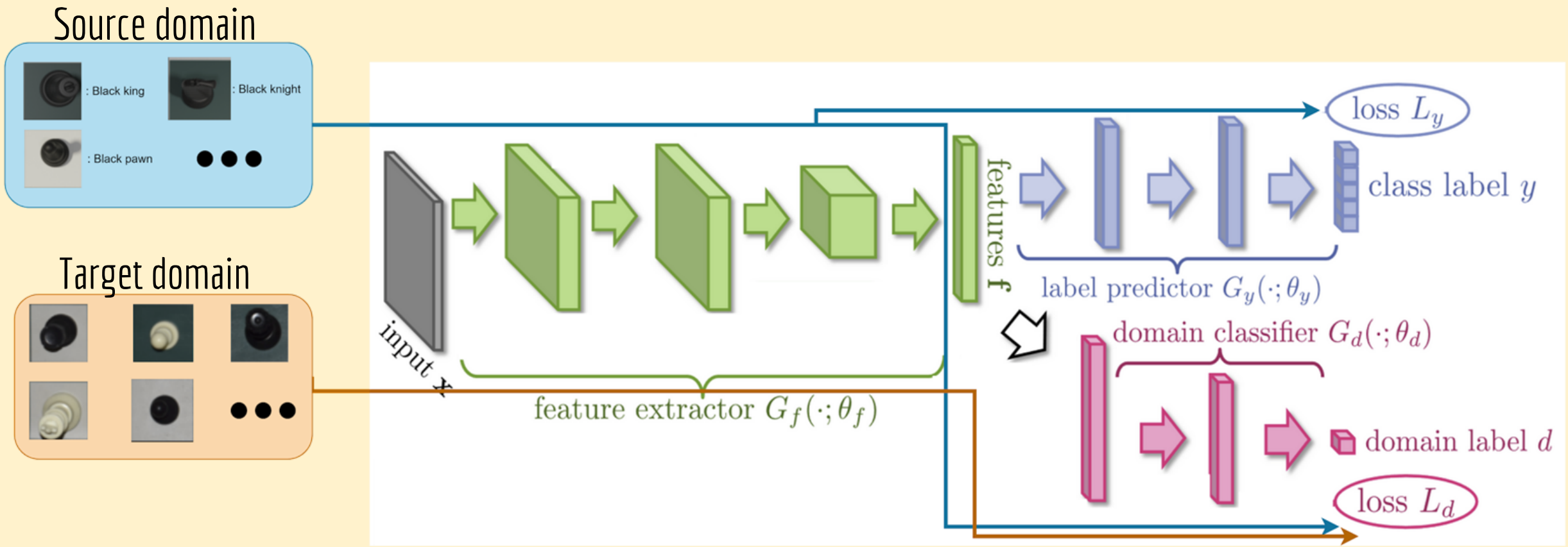
During training



[9] Y. Ganin *et al.*, 'Domain-Adversarial Training of Neural Networks', *arXiv [stat.ML]*. 2016.

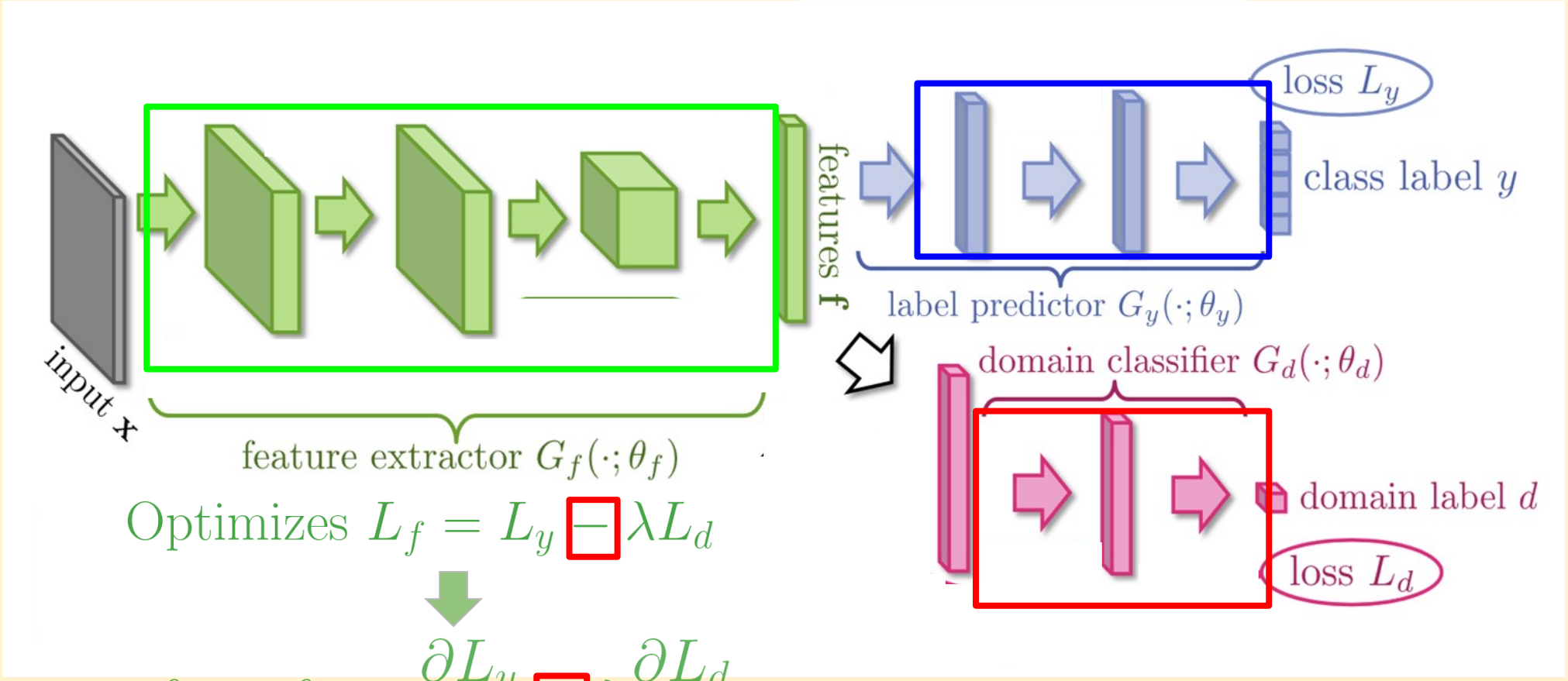
Architecture 3: Adversarial approach (DANN)

During training: Forward pass



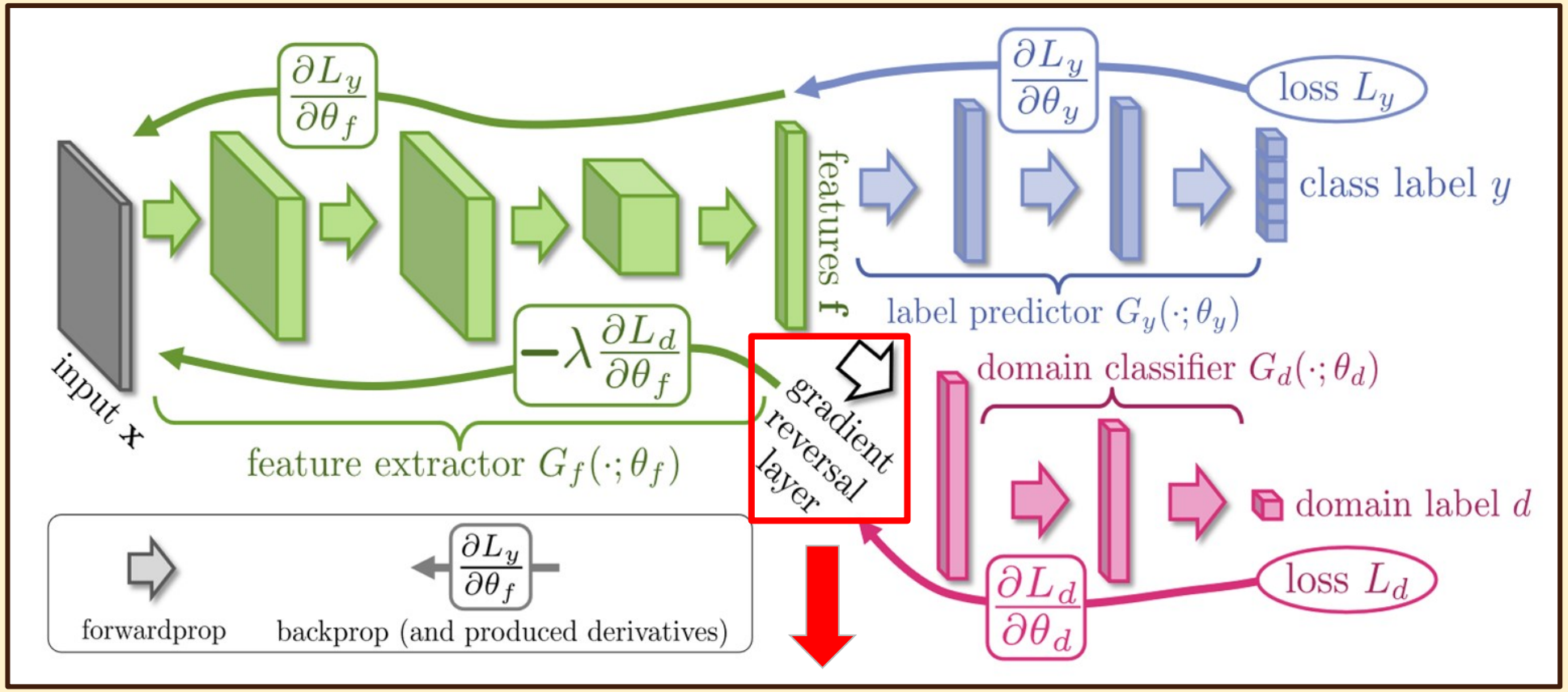
Architecture 3: Adversarial approach (DANN)

During training: Backward pass (1)



Architecture 3: Adversarial approach (DANN)

During training: Backward pass (2)



$$\theta_f = \theta_f - \frac{\partial L_y}{\partial \theta_f} + \lambda \frac{\partial L_d}{\partial \theta_f}$$

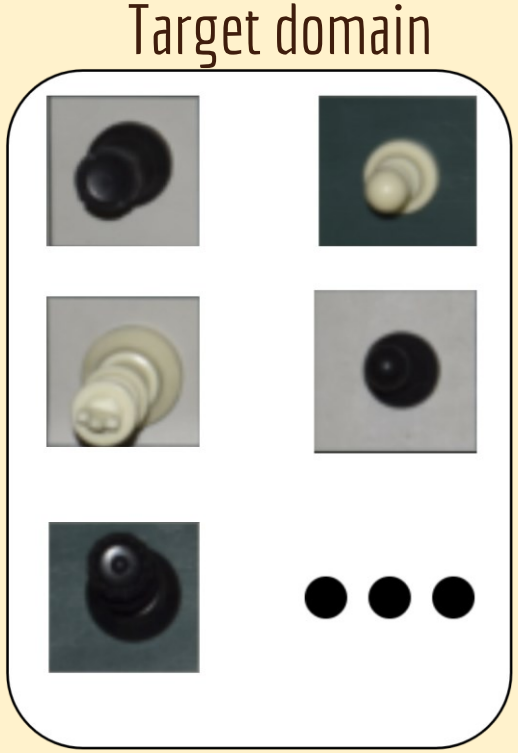
Gradient reversal layer:

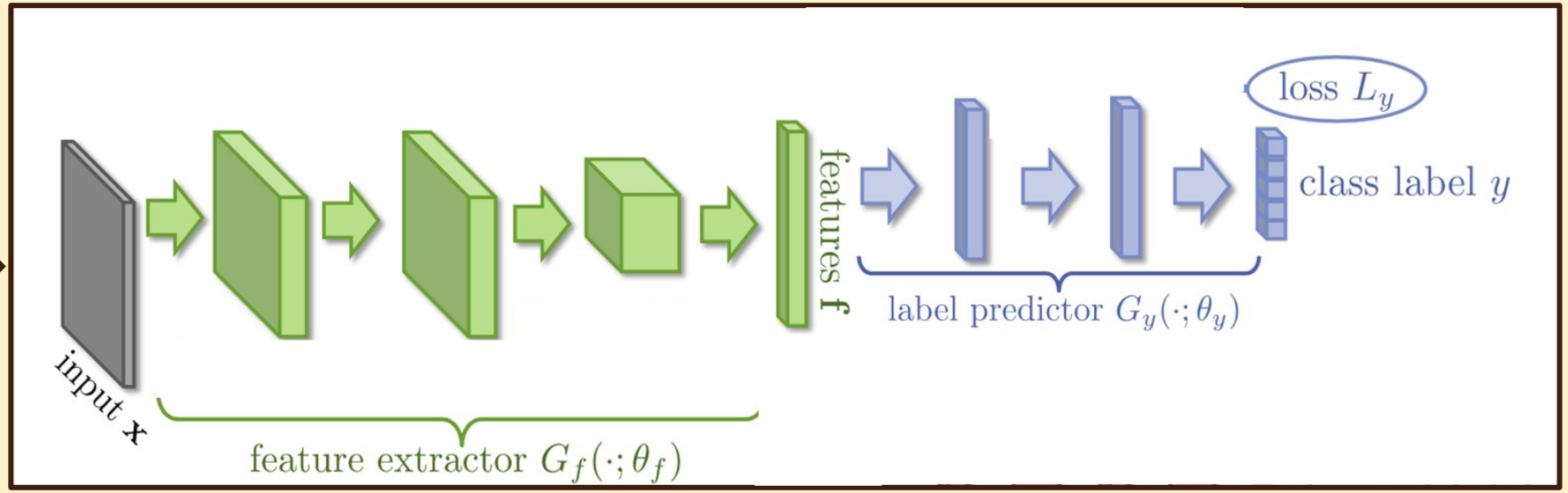
- Identity function during forward pass
- Multiplies by -λ during the backward pass

Architecture 3: Adversarial approach (DANN)

During testing

Target domain





Note: The domain discriminator can be discarded during testing

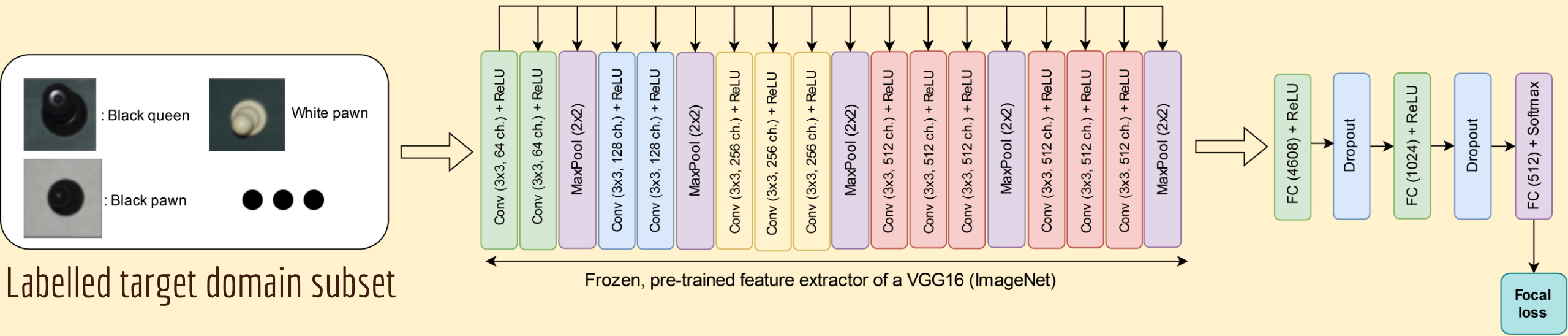
Architecture 4: VGG16 trained on a labelled target domain subset of the validation data (BASE-target)

We used a labelled validation subset of 3200 target domain square images

But what if those were enough to train the model without DA?

Here, we split the full validation set into 80% training and 20% validation

We do not use the source domain dataset



Labelled target domain subset

Hyperparameter tuning results - Base Models

Gamma	Dropout Rate	Best Validation Accuracy
2	0	59.7%
2	0.2	61.4%
2	0.5	56.7%
5	0	56.6%
5	0.2	56.9%
5	0.5	58.4%

Base-source

Gamma	Dropout Rate	Best Validation Accuracy
2	0	94.7%
2	0.2	93.3%
2	0.5	93.8%
5	0	94.7%
5	0.2	94.9%
5	0.5	93.8%

Base-target

Hyperparameter tuning results - DA models

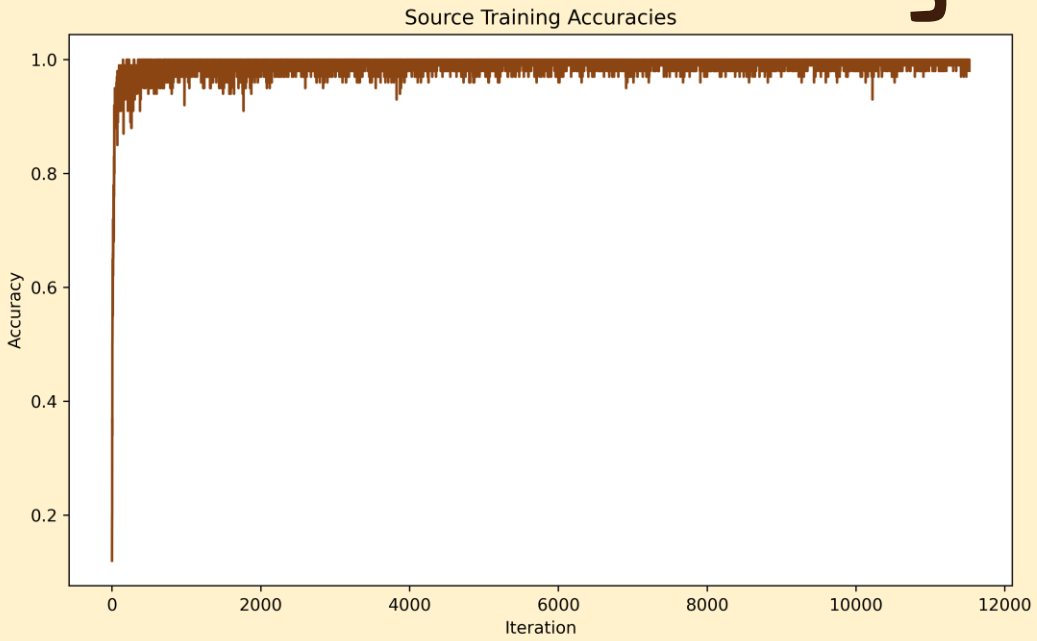
Gamma (Focal Loss)	Learning Rate	Lambda Max (DA factor)	Best Validation Accuracy
2	0.001	0.1	80.9%
2	0.001	1	83.2%
2	0.001	10	74.7%
2	0.0005	0.1	80.7%
2	0.0005	1	83.6%
2	0.0005	10	84.2%
5	0.001	0.1	81.1%
5	0.001	1	81.4%
5	0.001	10	83.5%
5	0.0005	0.1	82.5%
5	0.0005	1	85.4%
5	0.0005	10	84.4%

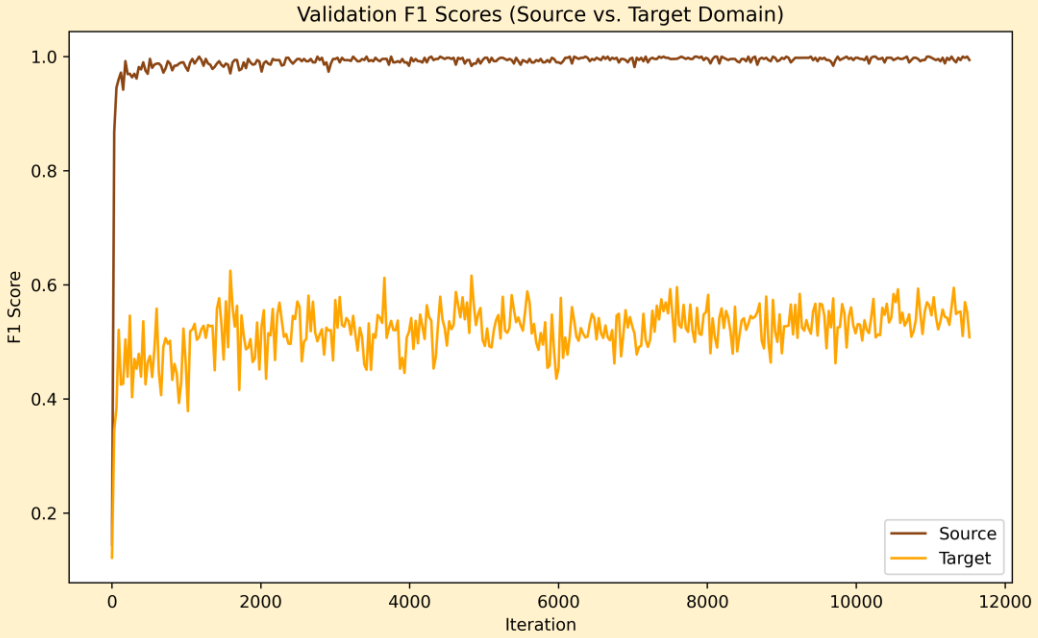
CORAL

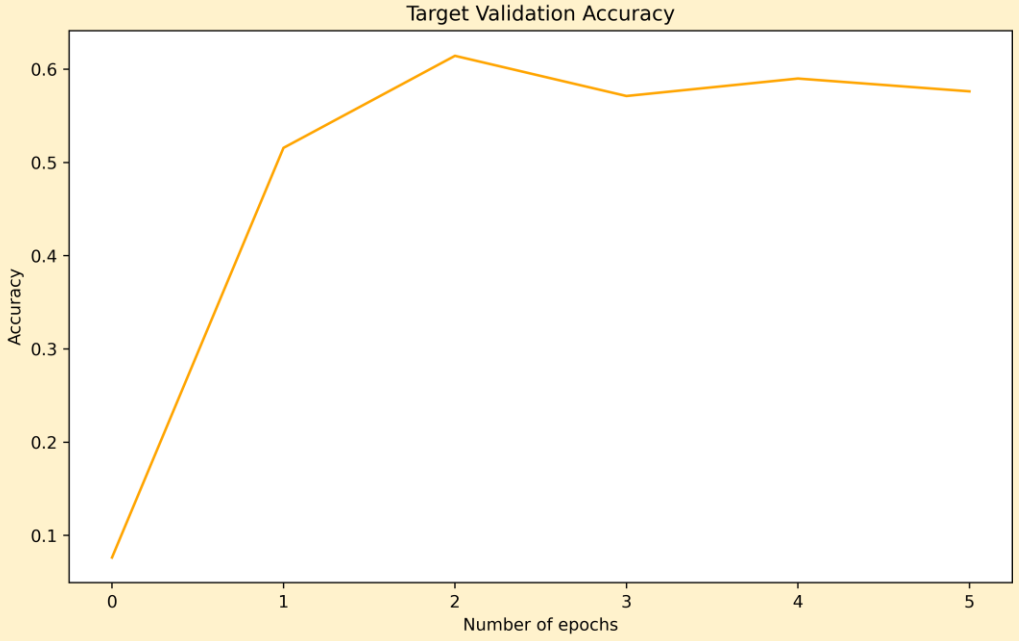
Gamma (Focal Loss)	Learning Rate	Lambda (DA factor)	Best Validation Accuracy
2	0.02	0.2	82.3%
2	0.02	variable	82.4%
2	0.02	0.3	84.4%
2	0.01	0.2	84.2%
2	0.01	variable	82.6%
2	0.01	0.3	78.8%
5	0.02	0.2	84.8%
5	0.02	variable	79.9%
5	0.02	0.3	85.0%
5	0.01	0.2	78.9%
5	0.01	variable	77.7%
5	0.01	0.3	76.5%

DANN

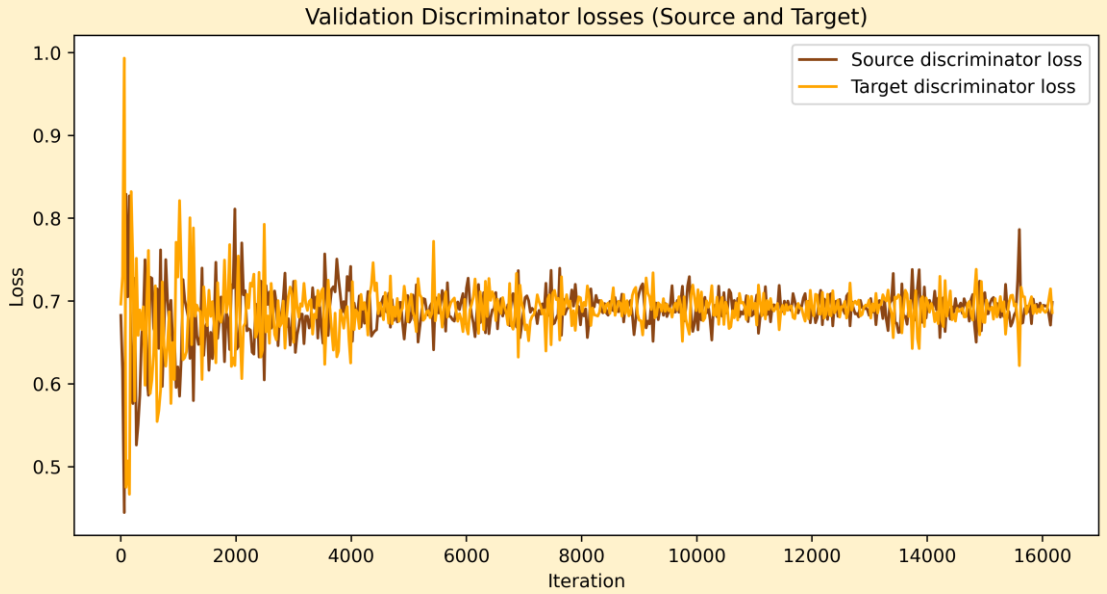
Convergence: BASE-source

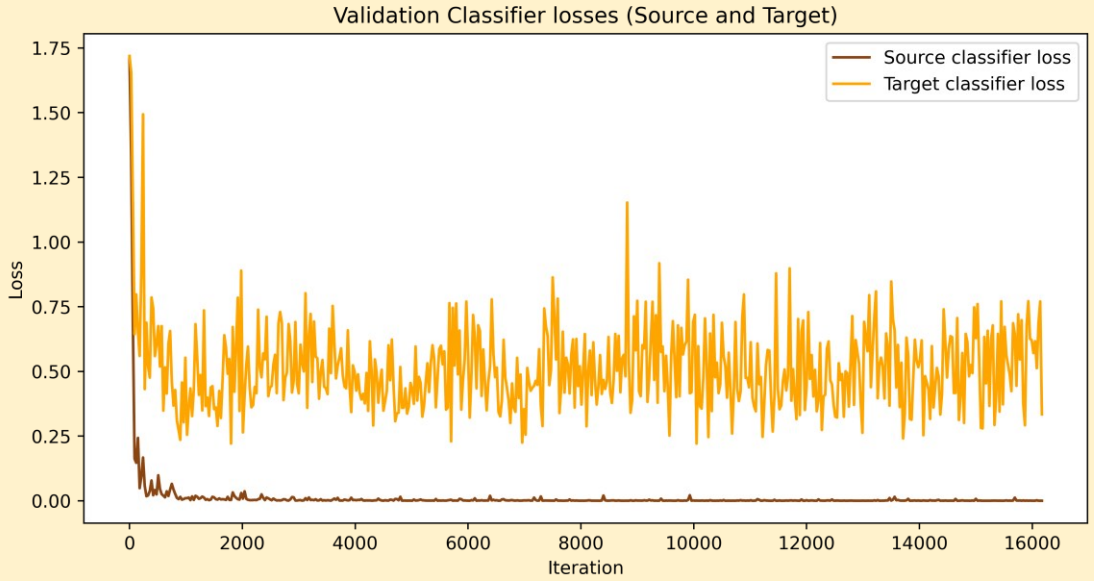


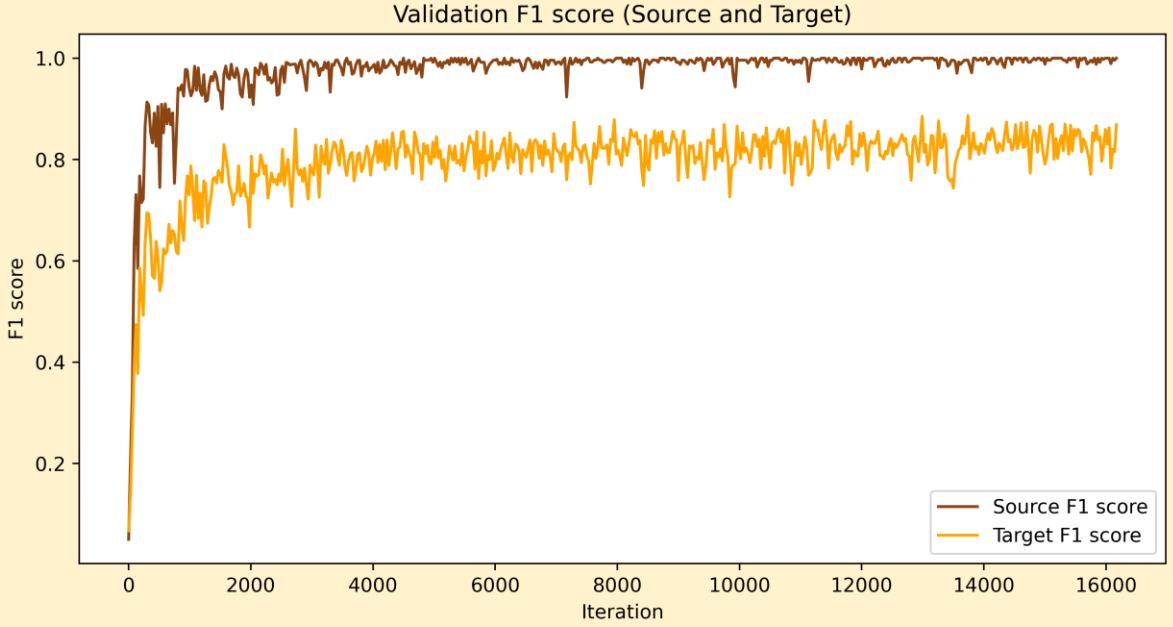


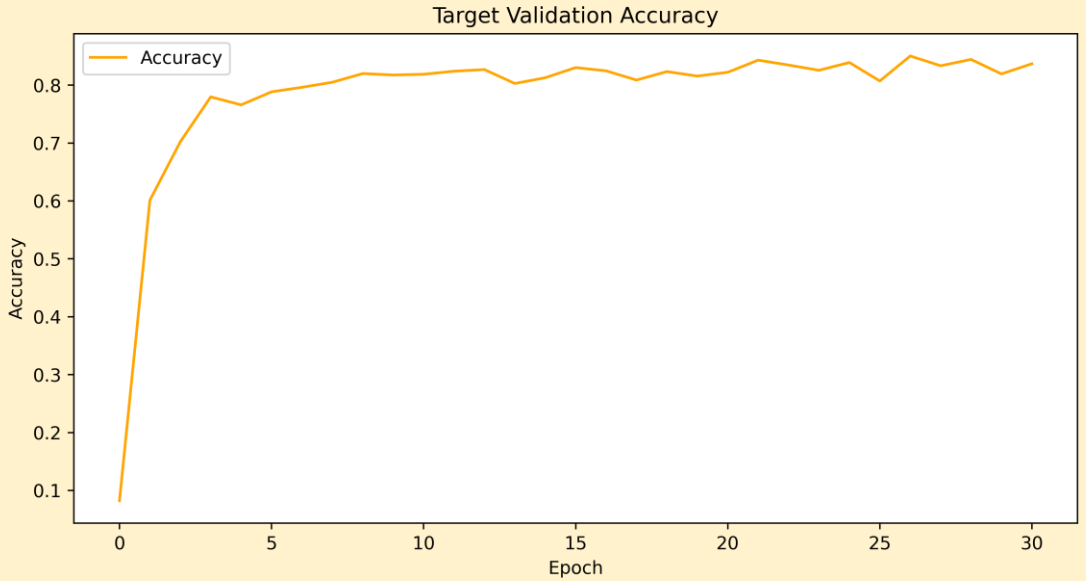


Convergence: DANN

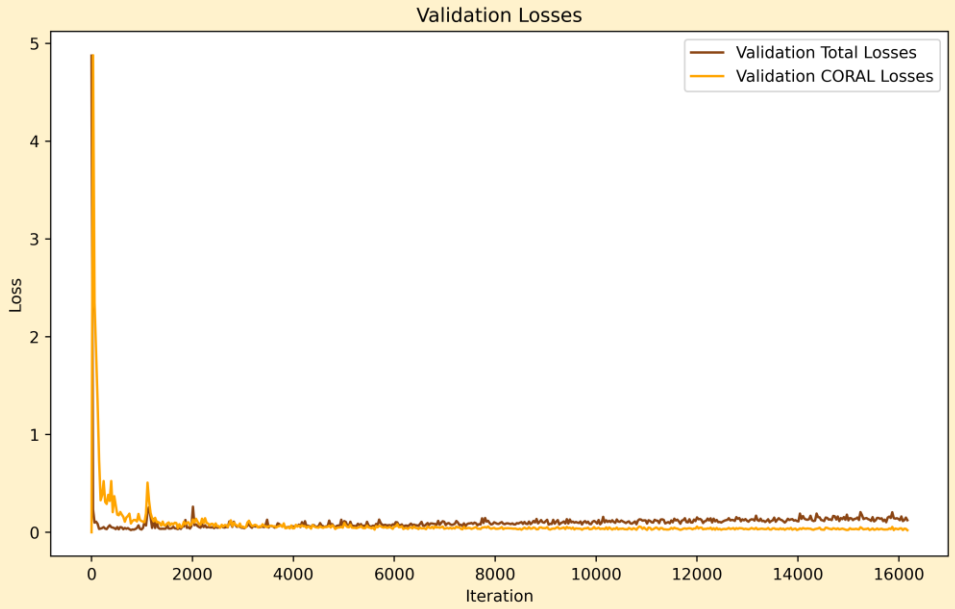


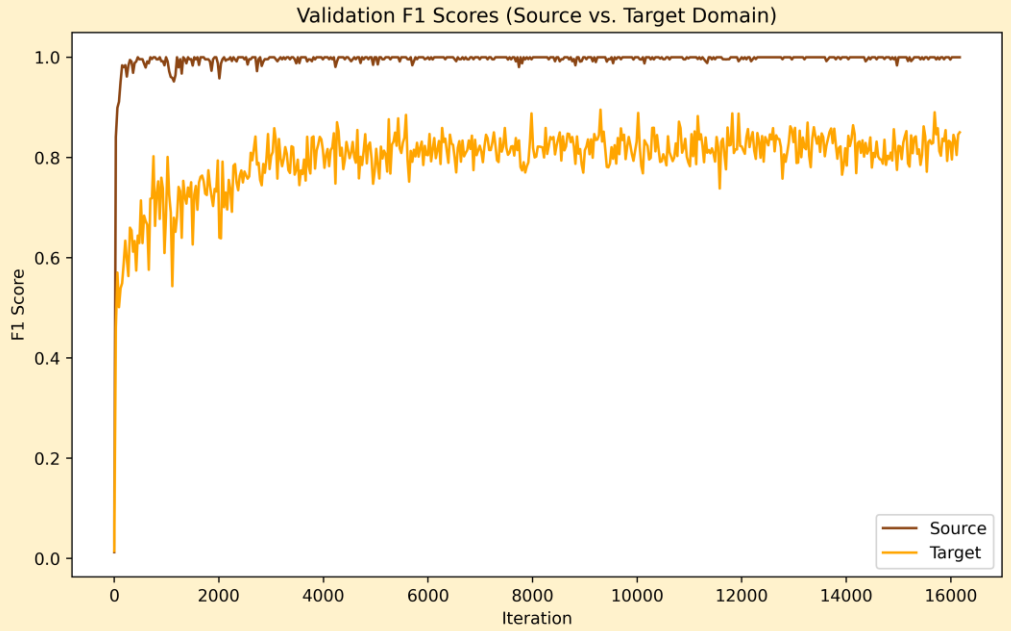


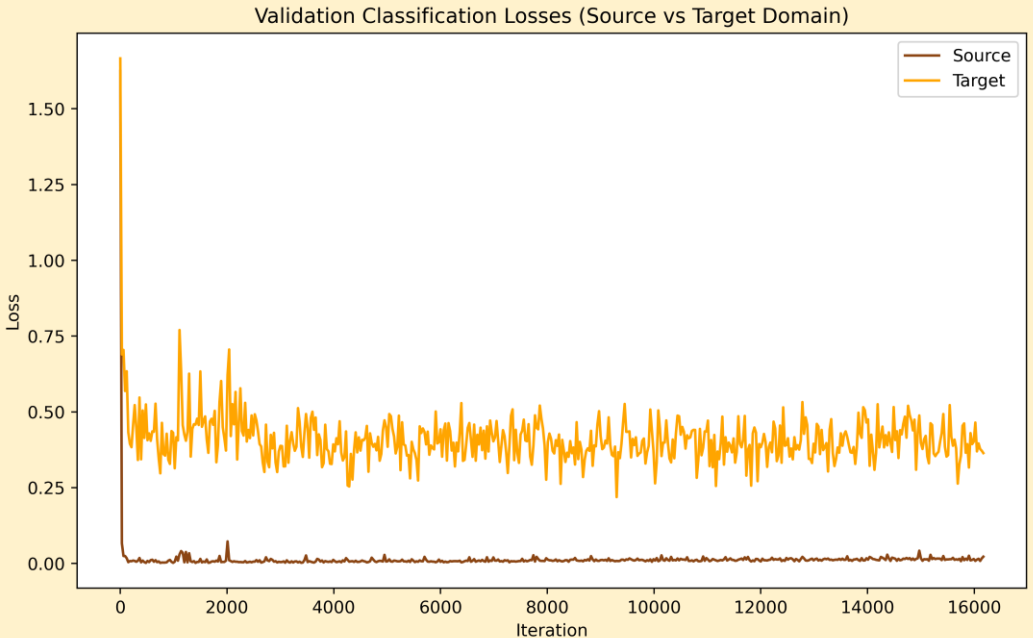


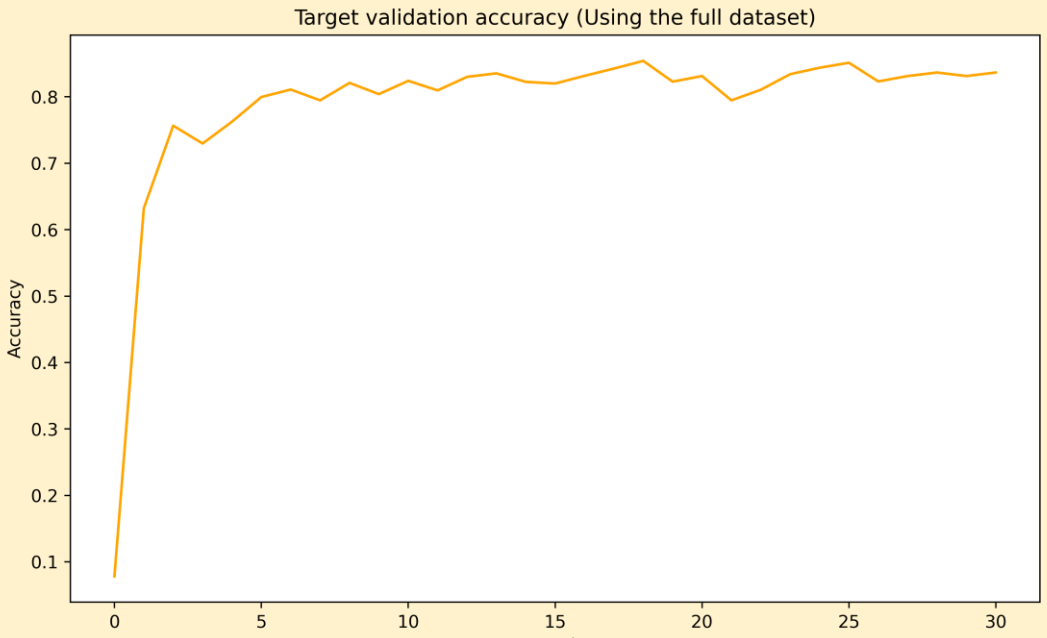


Convergence: CORAL

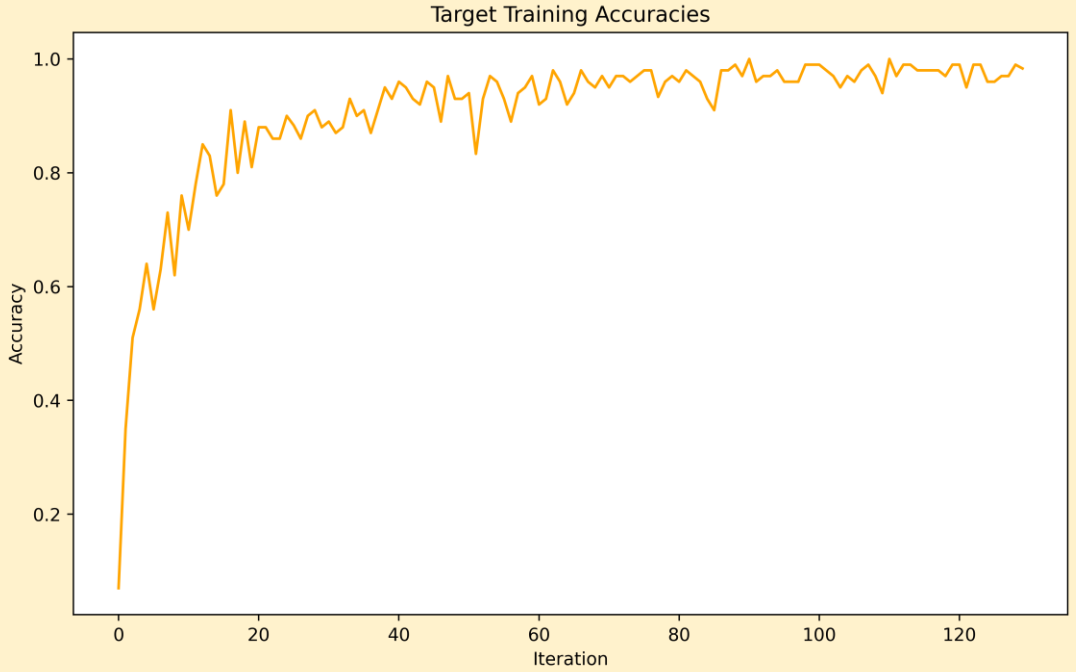


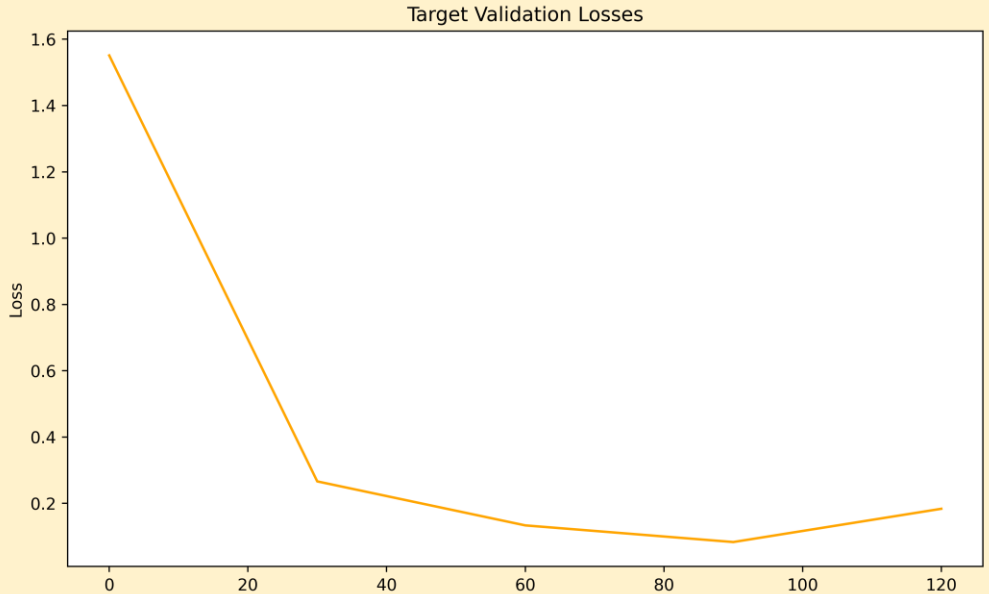


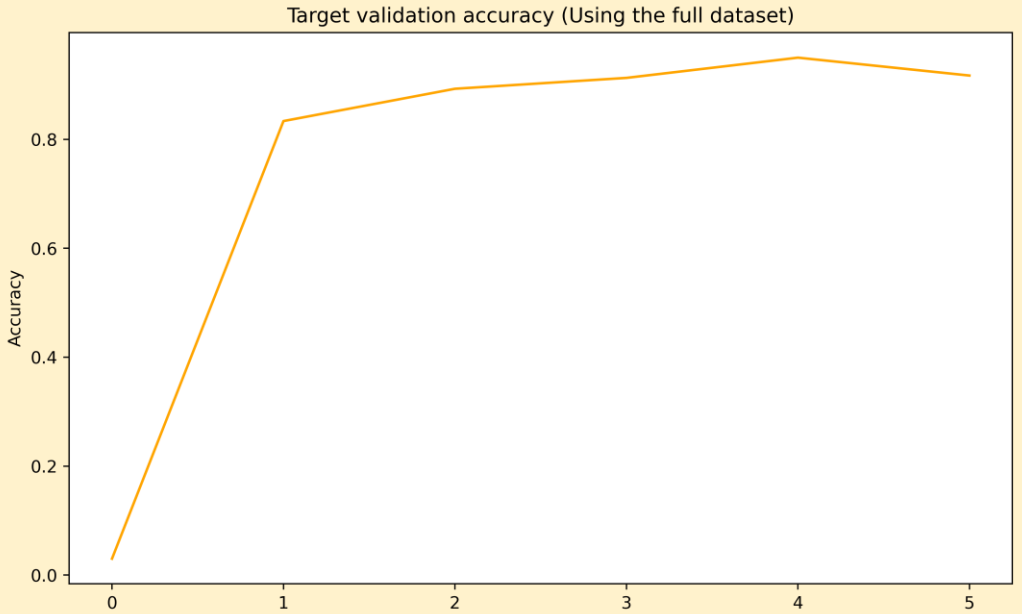




Convergence: BASE-target







Results: Testing metrics on balanced source domain

Model	Testing Accuracy	F1-Score	Average AUPRC
BASE-source	99.17%	0.992	0.992
CORAL	99.94%	0.999	0.999
DANN	99.68%	0.997	0.997
BASE-target	36.56%	0.340	0.429

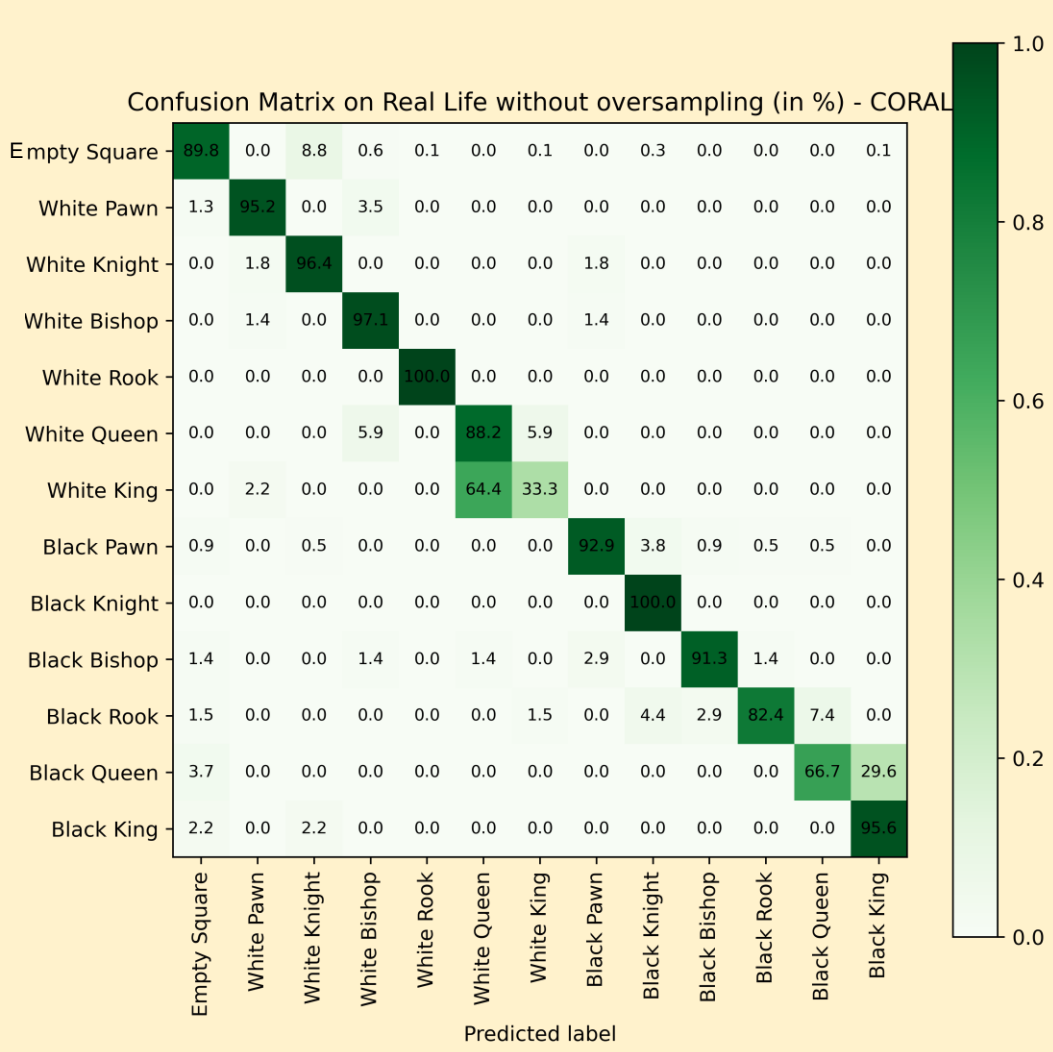
Results: Testing metrics on balanced target domain

Model	Testing Accuracy	F1-Score	Average AUPRC
BASE-source	57.59%	0.553	0.595
CORAL	85.43%	0.852	0.859
DANN	83.59%	0.832	0.842
BASE-target	93.00%	0.930	0.937

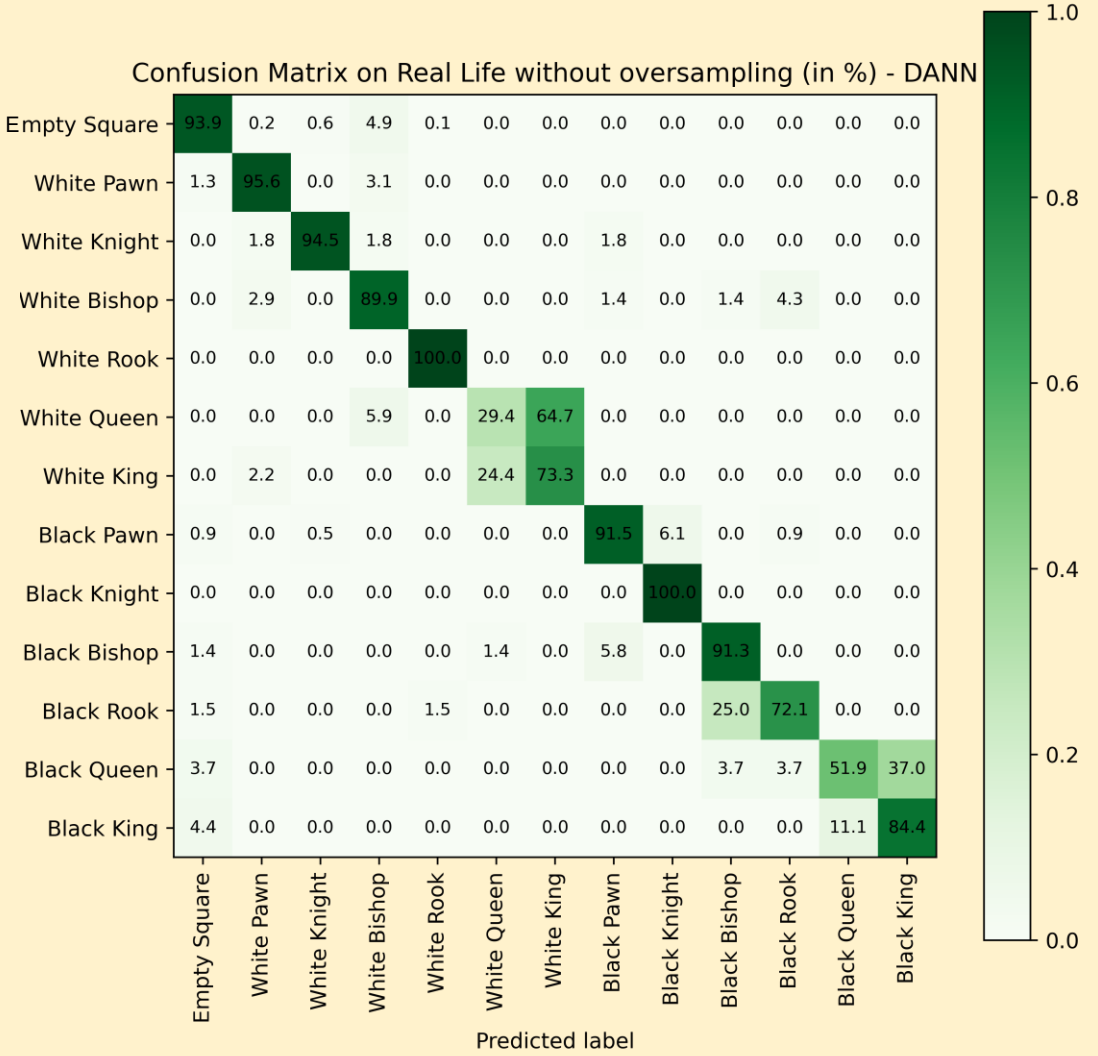
Results: Testing metrics on imbalanced target domain

Model	Testing Accuracy	F1-Score	Average AUPRC
BASE-source	87.43%	0.874	0.556
CORAL	89.91%	0.919	0.825
DANN	92.31%	0.932	0.793
BASE-target	95.18%	0.956	0.889

Results: Confusion matrices on imbalanced target domain

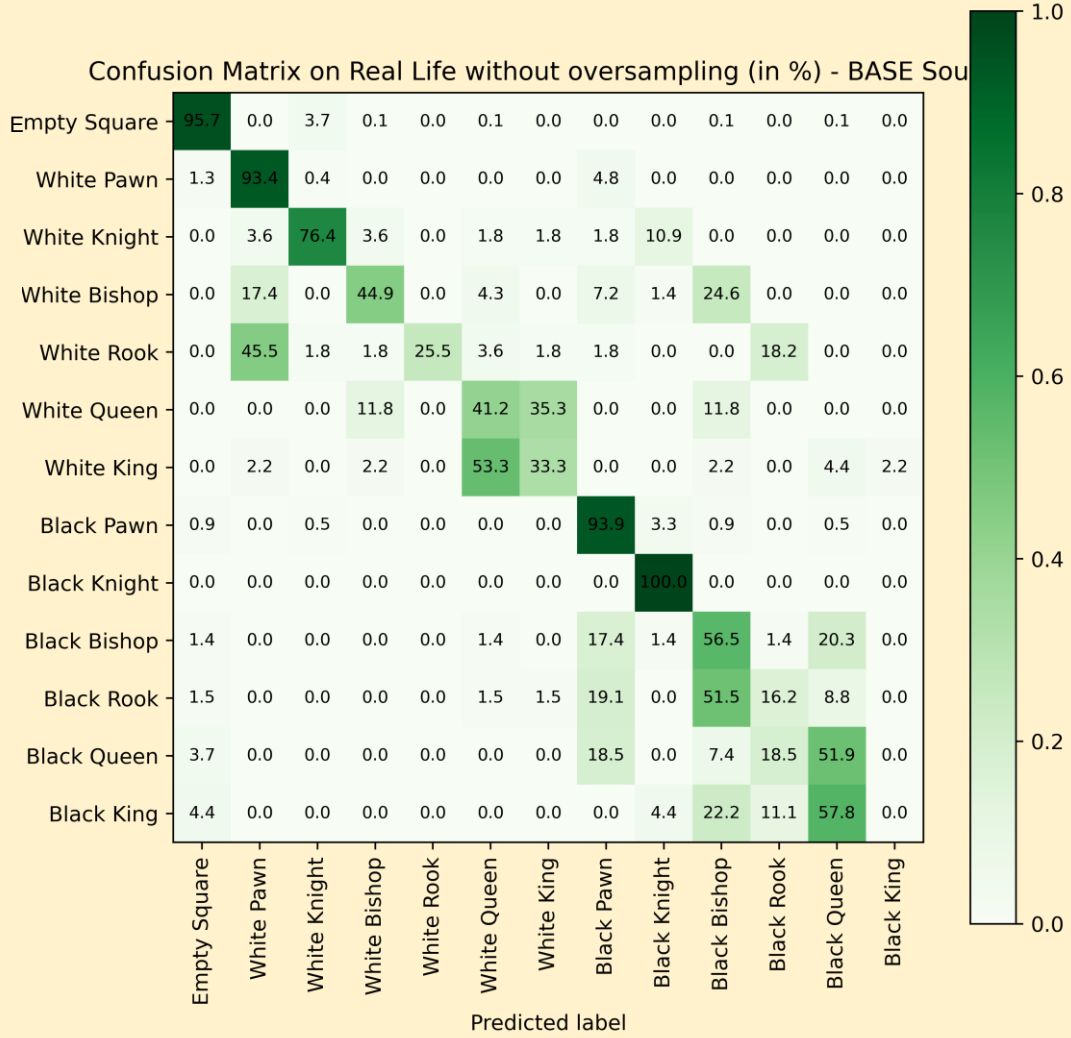


CORAL

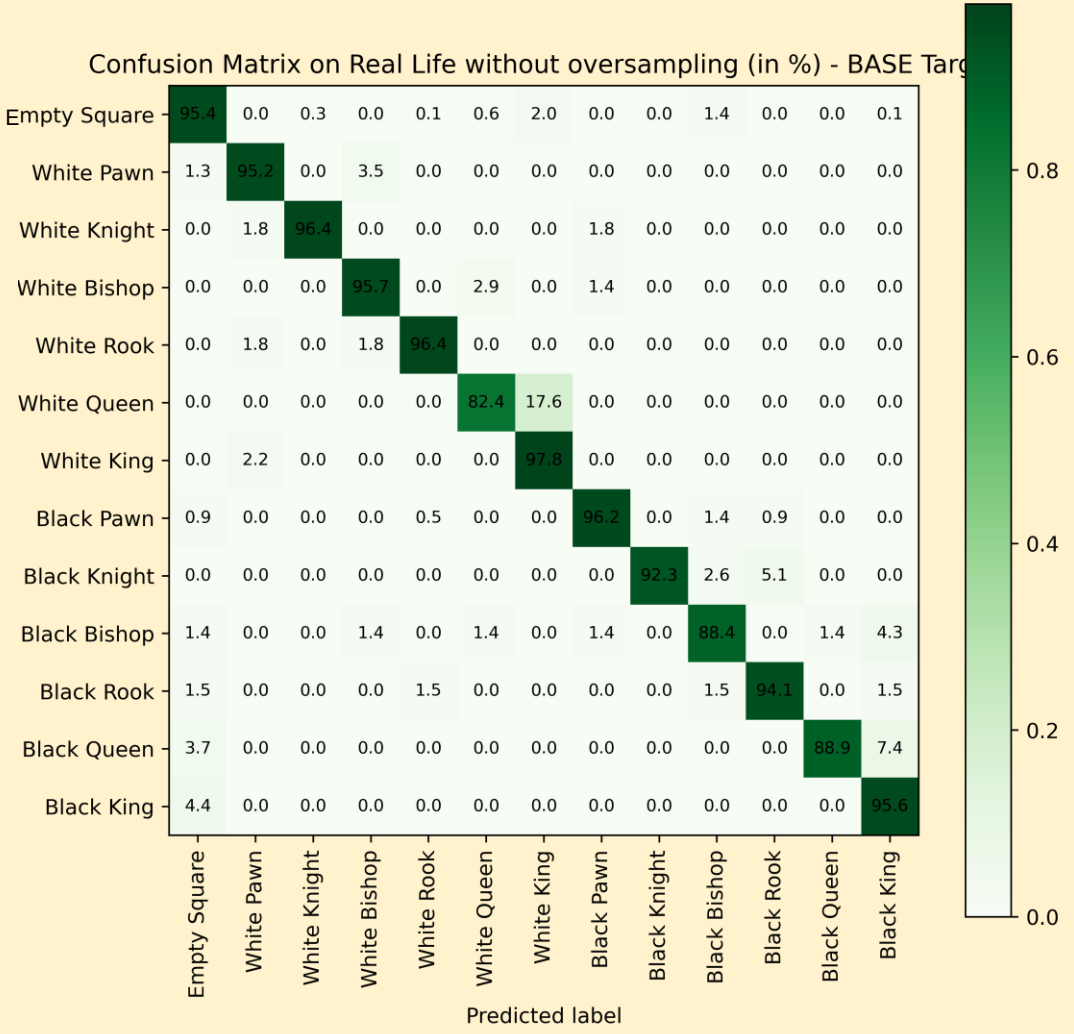


DANN

Results: Confusion matrices on imbalanced target domain



BASE-source



BASE-target

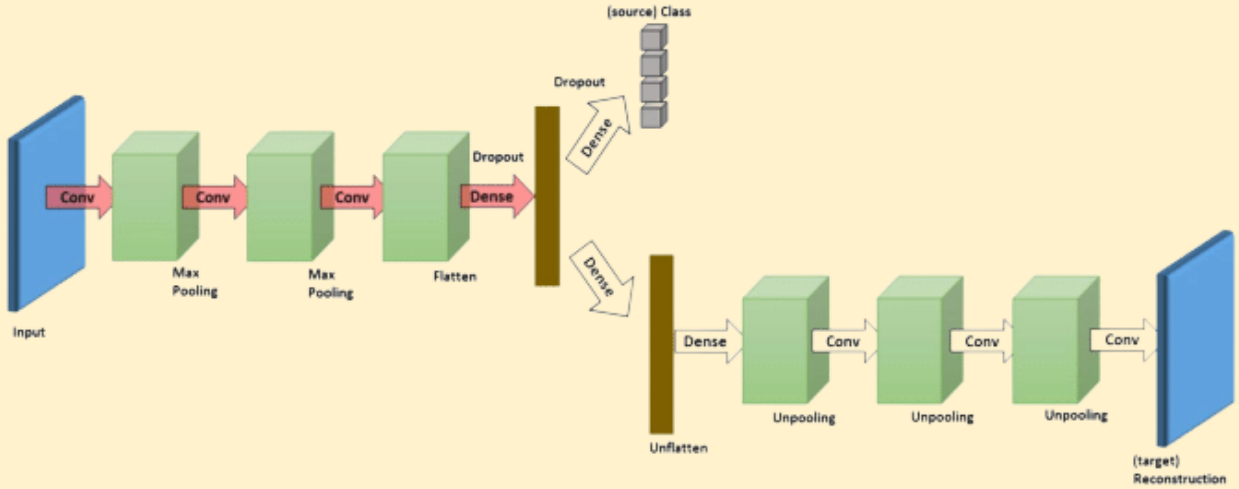
Conclusions

- DANN & CORAL achieved results comparable to BASE-target
- Acceptable results without requiring data labelling (For the most part)
- Abundance of source domain data
- Flexibility in manually tuning the feature & target distributions

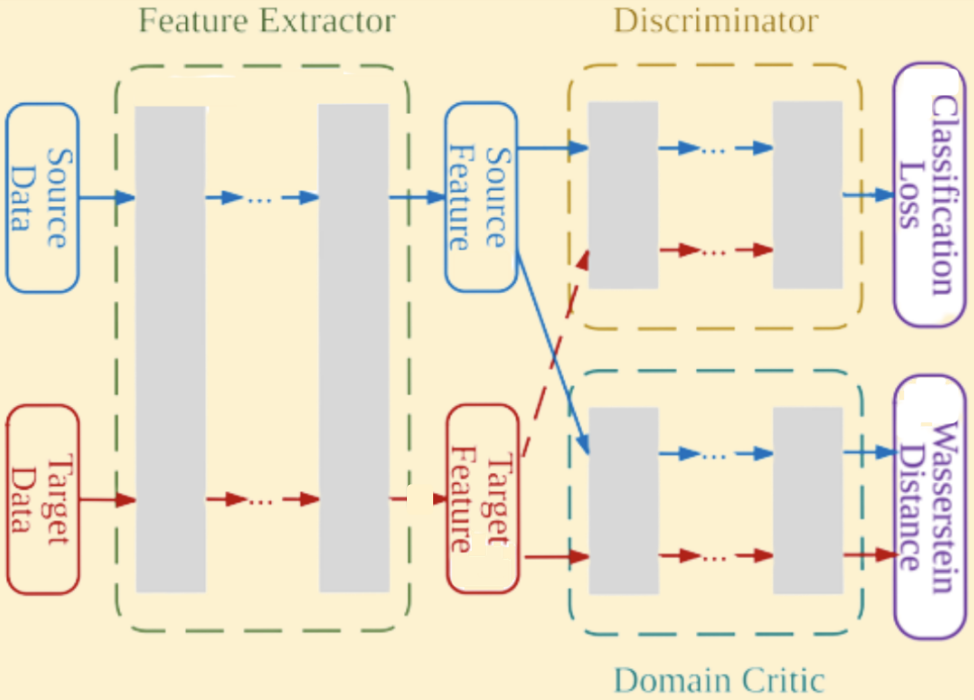
- Data generation was not straightforward at first and took time to R&D
- Hard time distinguishing kings & queens, a downside of using the top-view
- Not invariant to piece set or board texture
- Does not utilize the rules of chess as natural prediction constraints

Other possible approaches

Alternative 1: Deep-Reconstruction Classification Network (DRCN)[10]



Alternative 2: Wasserstein Distance Guided Representation Learning Model (WDGRL)[11]



[10]: M. Ghifary et al., "Deep reconstruction-classification networks for unsupervised domain adaptation," in Proc. Eur. Conf. Comput. Vis. (ECCV), pp. 597-613, 2016.
[11]: J. Shen et al., "Wasserstein distance guided representation learning for domain adaptation," in Proc. AAAI Conf. Artif. Intell., vol. 32, no. 1, 2018.

References

[1] S. Ben-David et al., "A theory of learning from different domains," Mach. Learn., vol. 79, pp. 151-175, 2010.

[2] A. D. S. D. Neto and R. M. Campello, "Chess position identification using pieces classification based on synthetic images generation and deep neural network fine-tuning," in Proc. 21st Symp. Virtual Augmented Reality (SVR), pp. 152-160, IEEE, Oct. 2019.

[3] Elucidation, "Elucidation/chessboarddetect: Hodgepodge of chessboard detection algorithms on images from actual matches.," GitHub. [Online]. Available: https://github.com/Elucidation/ChessboardDetect. [Accessed: 10-Apr-2023].

[4] "Fen to image," Chessvisionai RSS, https://chessvision.ai/docs/tools/fen2image/ (accessed Jun. 1, 2023).

[5] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, 'Focal Loss for Dense Object Detection', arXiv [cs.CV]. 2018.

[6] B. Sun, J. Feng, and K. Saenko, 'Correlation Alignment for Unsupervised Domain Adaptation', CoRR, vol. abs/1612.01939, 2016.

[7] K. Simonyan and A. Zisserman, 'Very Deep Convolutional Networks for Large-Scale Image Recognition', CoRR, vol. abs/1409.1556, 2014.

[8] B. Sun, J. Feng, and K. Saenko, 'Correlation Alignment for Unsupervised Domain Adaptation', CoRR, vol. abs/1612.01939, 2016.

[9] Y. Ganin et al., 'Domain-Adversarial Training of Neural Networks', arXiv [stat.ML]. 2016.

[10]: M. Ghifary et al., "Deep reconstruction-classification networks for unsupervised domain adaptation," in Proc. Eur. Conf. Comput. Vis. (ECCV), pp. 597-613, 2016.

[11]: J. Shen et al., "Wasserstein distance guided representation learning for domain adaptation," in Proc. AAAI Conf. Artif. Intell., vol. 32, no. 1, 2018.

Questions?