EXPLORING CONVOLUTIONAL NEURAL NETWORKS TRAINING STRATEGIES FOR PILE UP CORRECTION IN SINGLE PARTICLE COUNTING

T. Habermann*, Fulda University of Applied Sciences, Fulda, Germany
M. Hamdan†, Cardiff University, Cardiff, United Kingdom

Abstract

This work investigates the effect of different training strategies on the performance of convolutional neural networks (CNNs) for real-time pile-up identification and correction in single particle counters at the GSI Helmholtz Centre for Heavy Ion Research. Building on a previously developed CNN capable of detecting particle pulses without domain-specific knowledge, we examine the influence of different loss functions and their hyperparameters on the network's ability to accurately localize overlapping events. We also propose new metrics for particle counting and show how to adapt common metrics for precision and recall to allow a user-defined localization tolerance. Using a dataset of approximately 26,000 manually labeled pulses, we analyze how these training choices impact particle counting and localization performance. The findings provide insights for developing particle counting systems suitable for real-time applications, including potential implementation on FPGA hardware.

INTRODUCTION

Accurate particle counting and precise arrival time determination are fundamental requirements in experimental nuclear and particle physics, as well as in accelerator beam diagnostics. In high-rate environments such as those encountered at the GSI Helmholtz Centre for Heavy Ion Research, scintillator-based single particle counters frequently suffer from pile-up events, where multiple particles arrive within short time intervals and produce superimposed pulse shapes at the detector output. Typically, the number of pile-ups in extracted beams from particle accelerators is much more than one expects from a random discrete process or Poisson process [1, 2].

Previous work [3] demonstrated how convolutional neural networks (CNNs) can be used for pile-up correction in single particle counting and outperform traditional thresholding and local maxima methods without relying on domain-specific knowledge or explicit analytical pulse models. Tested on a dataset of approximately 26,000 manually labeled events, the proposed architecture achieved accurate particle counting and was designed with real-time FPGA implementation in mind. These results established the feasibility of machine-learning-based pile-up correction for accelerator applications.

* tobias.habermann@informatik.hs-fulda.de
† mahdie.hamdan@outlook.com

However, while the network architecture was optimized for deployment constraints, the influence of the training strategy on detection behavior was not systematically investigated. In particular, because particle counting involves sparse event localization in highly imbalanced time-series data, the loss function must address this class imbalance and prioritize particle counting. In [3], the focal loss function [4] was chosen as it is a well-known and widely used loss function for tasks with class imbalance. But a proper investigation into loss functions is still missing.

In this work, we therefore investigate the impact of different loss functions and hyperparameter settings on CNN performance for real-time pile-up identification. The results provide practical guidance for the development of reliable particle counting systems and show how to improve the approach for localization and particle counting.

BACKGROUND

The CNN model presented in [3] is processing windows of data in sequential order. Each window of data contains 256 data points. Figure 1 shows a window of data together

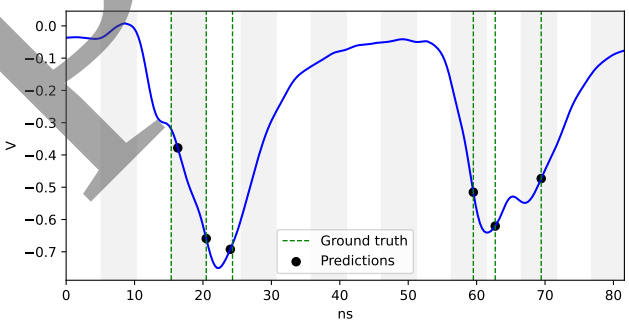

Figure 1: An example containing 6 pulses with all true peak locations and predicted locations shown in the figure. Also, each cell is highlighted.

with the predicted peak positions shown as black dots. The correct peak positions (ground truth) are marked with dotted green lines.

The CNN model itself splits each window into cells containing 16 data points each. The gray and white background in Fig. 1 is used to highlight each cell. Each cell can predict up to m peaks, and each peak prediction is composed of two features, which are the confidence score and the relative position in the cell. Both features are normalized between zero and one using a sigmoid activation function. A peak is predicted when the confidence score is higher than 0.5. This concept is similar to YOLO [5] box detection models. The CNN model architecture from [3] is a simple CNN with

just 6 layers. The architecture is shown in Table 1, where it can be seen that the stride settings of the first two layers define the cell size, and the kernel size of the following layers define how many neighboring cells each cell can take into consideration for peak prediction.

Table 1: Model Architecture for n Cells Where Each Cell Can Detect up to m Pulses

Layer	Kernel	Stride	Filters	Output shape
input	-	-	-	$(16n, 1)$
conv1	5	4	8	$(4n, 8)$
conv2	5	4	16	$(n, 16)$
conv3	3	1	16	$(n, 16)$
conv4	1	1	64	$(n, 64)$
conv5	3	1	32	$(n, 32)$
conv6	1	1	$2m$	$(n, 2m)$
reshape	-	-	-	$(n, m, 2)$

LOSS FUNCTIONS

We inspect two loss functions as alternatives to the focal loss [4] used in [3]. In our setting, each prediction window contains far more background outputs than true particle positions, creating a strong class imbalance that plain cross-entropy cannot handle well [4]. While the focal loss is an established remedy, it is static — its weighting does not adapt as training progresses — and it operates purely pointwise, without any notion of global detection overlap. We therefore consider the **Gradient Harmonizing Mechanism** (GHM) [6], which remedies the static nature of focal loss by adaptively reweighting examples based on their gradient distribution, and the **Unified Focal Loss** (UFL) [7], which combines pointwise focal modulation with a region-based Tversky overlap term.

The Gradient Harmonizing Mechanism (GHM)

For a predicted probability $p \in [0, 1]$ and binary ground truth $p^* \in \{0, 1\}$, define $g = |p - p^*|$. One can verify that g equals the norm of the gradient of the binary cross-entropy loss w.r.t. the model output, making it a natural scalar measure of example difficulty.

The key insight of GHM [6] is that the *distribution* of g over a training batch is highly non-uniform. Easy negatives cluster densely near $g \approx 0$ and, despite contributing little individually, overwhelm the gradient in aggregate. A smaller cluster of hard outliers near $g \approx 1$ can further destabilize training. The focal loss suppresses the easy majority, but its weighting is static and ignores the outlier problem. GHM addresses both by weighting each example inversely to how many other examples share a similar gradient norm — densely populated regions are down-weighted, sparse ones are not.

In practice, computing exact neighbor counts is $O(N^2)$. Instead, the unit interval can be partitioned into M equal bins of width $1/M$, and R_j denotes the number of examples whose gradient norm falls in bin j. Example i is assigned to bin $j = \lfloor g_i \cdot M \rfloor$, and its weight is set to N/R_j. The GHM classification loss is then:

$$L_{\text{GHM-C}} = \frac{1}{N} \sum_{i=1}^{N} \frac{N}{R_{j(i)}} L_{CE}(p_i, p_i^*),$$

where $j(i)$ denotes the bin of example i. The weights are normalized per batch for stability. Crucially, unlike focal loss, GHM is *adaptive*: the bin counts R_j are recomputed each mini-batch, so the weighting automatically tracks the evolving training distribution without any manual tuning.

Unified Focal Loss

The Unified Focal Loss (UFL) [7] unifies distribution-based losses (such as the focal loss), which reweight individual predictions to suppress the easy-example majority, with region-based losses (such as the Dice loss), which optimize the overlap between predicted and true positive sets and are inherently more robust to class imbalance, but blind to the difficulty of individual examples. The UFL combines both.

Starting from the focal loss,

$$\mathcal{L}_F = -\alpha(1 - p_t)^\gamma \log(p_t), \tag{1}$$

where $p_t = p$ if $p^* = 1$ and $p_t = 1 - p$ otherwise, the region-based counterpart is derived by considering the Dice loss,

$$\mathcal{L}_{DSC} = 1 - \frac{2TP}{2TP + FP + FN},$$

where TP, FP, FN are true positives, false positives, and false negatives. Since Dice treats FP and FN symmetrically, the **Tversky index** generalizes it with separate weights,

$$TI = \frac{TP}{TP + \delta\, FP + (1 - \delta)\, FN}, \tag{2}$$

giving Tversky loss $\mathcal{L}_T = 1 - TI$. Setting $\delta < 0.5$ penalizes missed detections more than false positives, which is natural for sparse particle counting. Applying a focal-style exponent then yields the **Focal Tversky loss**,

$$\mathcal{L}_{FT} = (1 - TI)^\gamma, \tag{3}$$

which further emphasizes windows where the model's overlap with the ground truth is poor.

The UFL takes a weighted combination $\mathcal{L}_{UF} = \lambda\, \mathcal{L}_F + (1 - \lambda)\, \mathcal{L}_{FT}$. A naive combination would carry six hyperparameters; the UFL reduces these to three by coupling the focal exponents of the two terms to a shared γ (as $(1 - \gamma)$ and γ respectively) and reusing δ from Eq. (2) as the class-weighting parameter in Eq. (1):

$$\mathcal{L}_{UF} = -\lambda\delta(1 - p_t)^{1-\gamma} \log(p_t) + (1 - \lambda)\left(1 - \frac{TP}{TP + \delta\, FP + (1 - \delta)\, FN}\right)^\gamma.$$

Here δ controls the FP/FN tradeoff, γ governs focal modulation strength for both components simultaneously, and λ blends the pointwise and region-based terms.

METRICS

In [3], the absolute difference between the number of predicted peaks and the number of true peaks was taken to measure the quality of the approach.

However, since the model predicts peaks on a per-window basis, this metric fails to penalize under- or overcounting within a given window. To address this limitation, we introduce the strict count metric (SC), where we count the number of windows where the model predicted the true number of peaks exactly and take the average as a percentage.

To measure the localization error of the model, [3] measured the distance of each peak prediction in a window to the closest candidate in the ground truth data. To prevent ambiguity, the localization error was only measured when the number of predicted peaks was equal to the number of real peaks.

Rather than matching the closest candidates, we propose to map predicted peak locations and actual peak locations onto a 1D image of a certain resolution. For example, with a window size of 8, when the model predicts three peaks at position 0, 4, 4 and the actual peaks are at 4, 5, the predicted and actual peaks can be mapped onto a 1D pixel image $p_{\text{map}} = (1, 0, 0, 0, 2, 0, 0, 0)$ and $y_{\text{map}} = (0, 0, 0, 0, 1, 1, 0, 0)$. This allows counting the true/false positives and negatives. So, for the example shown, there are two false positives, one true positive, and one false negative.

To allow a localization error of one pixel, every three pixels are summed up, lowering the resolution to $1/3$ of the window size, resulting in $p_{\text{map}} = (1, 2, 0)$ and $y_{\text{map}} = (0, 2, 0)$. This method allows for measuring the localization error, for a given tolerance, even when the predicted number of peaks does not match the number of actual peaks without ambiguity within a given error range. Also, as this metric relies on the confidence score and relative position predictions of the model, the metric provides a more comprehensive performance overview of the model. We then take the commonly used F1 score [8], which combines precision and recall, to measure localization performance with a tolerance of ϵ pixels

$$F_1(\epsilon) = \frac{2TP}{2TP + FP + FN}. \tag{4}$$

EXPERIMENTS

For a fair comparison of the loss functions, we tested all hyperparameter settings of each loss function and introduced a hyperparameter c for weighing between the position and confidence loss

$$\mathcal{L} = c\mathcal{L}_{Conf} + (1 - c)\mathcal{L}_{Pos}. \tag{5}$$

We repeated each experiment for each hyperparameter setting three times and searched for the best hyperparameter settings for each approach. The results for the focal loss function are used as a reference, as it was the loss function used in [3].

Particle Counting

For particle counting, we searched for the best *SC* score. Table 2 shows the best results for particle counting for each loss function. The best hyperparameters for focal loss were

Table 2: The Best Results for Particle Counting

Loss	$\mathbf{F_1(0)}$	$\mathbf{F_1(1)}$	SC	Diff.
Focal Loss [3]	0.9285	0.9865	98.57%	94
GHM Loss	0.8938	0.9813	98.24%	-229
Uni. F. Loss	0.9114	0.9787	**98.66%**	**-13**

$c = 0.9, \alpha = 0.5, \gamma = 1.0$, for GHM loss $c = 0.625$, and for unified focal loss $c = 0.5, \gamma = 0.5, \lambda = 0.5, \delta = 0.5$. It can be seen that the unified focal loss performs best with a strict count of 98.66%.

Particle Localization

Additionally, we searched for the best hyperparameters when prioritizing particle localization over particle counting. Table 3 shows the best results for the F_1 score for each loss function.

Table 3: The Best Results for Particle Localization

Loss	$\mathbf{F_1(0)}$	$\mathbf{F_1(1)}$	SC	Diff.
Focal Loss [3]	**0.9401**	**0.9904**	97.37%	301
GHM Loss	0.9357	0.987	97.55%	-445
Uni. F. Loss	0.9388	0.9887	97.11%	573

The best hyperparameters for focal loss were $c = 0.5, \alpha = 0.5, \gamma = 3.0$, for GHM loss $c = 0.03125$, and for unified focal loss $c = 0.25, \gamma = 2.0, \lambda = 0.5, \delta = 0.6$. It can be seen that the focal loss yields the best results when particle localization is prioritized.

When comparing the $F_1(1)$ score from Table 2 with Table 3, it can be seen that the localization performances are very similar to each other, although the counting performances, measured by *SC* and the total number of under-/overcounted peaks "Diff.", yield far better results when just prioritizing particle counting.

CONCLUSION

In this work, we evaluated different loss functions for CNNs used for pile-up correction in single particle counting. We determined the most fitting hyperparameter settings for each loss function to tune the CNN for prioritizing particle counting or particle localization. We also proposed new metrics to measure particle counting and localization performance. It was shown that the unified focal loss improves the particle counting performance of CNNs, surpassing the previous approach using focal loss.

ACKNOWLEDGMENTS

The data was collected at the HTP beamline at GSI Helmholtz Centre for Heavy Ion Research and the work was funded by the Federal Ministry of Research, Technology and Space, Germany under Grant 05K25REA.

REFERENCES

[1] R. Singh, P. Forck, and S. Sorge, "Reducing fluctuations in slow-extraction beam spill using transit-time-dependent tune modulation", *Phys. Rev. Appl.*, vol. 13, no. 4, p. 044076, Apr. 2020. doi:10.1103/PhysRevApplied.13.044076

[2] P. Niedermayer and R. Singh, "Excitation signal optimization for minimizing fluctuations in knock out slow extraction", *Sci. Rep.*, vol. 14, p. 10310, May 2024. doi:10.1038/s41598-024-60966-y

[3] T. Habermann, M. Kumm, and R. Singh, "Application of convolutional neural networks for pile up correction in single particle counting", in *Proc. IBIC 2025*, Liverpool, UK, pp. 152–155, Sep. 2025. doi:10.18429/JACoW-IBIC2025-MOPCO36

[4] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection", in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017. doi:10.1109/ICCV.2017.324

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016. doi:10.1109/CVPR.2016.91

[6] B. Li, Y. Liu, and X. Wang, "Gradient harmonized single-stage detector", in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 8577–8584, 2019. doi:10.1609/aaai.v33i01.33018577

[7] M. Yeung, E. Sala, C.-B. Schönlieb, and L. Rundo, "Unified focal loss: generalising dice and cross entropy-based losses to handle class imbalanced medical image segmentation", *Computerized Medical Imaging and Graphics*, vol. 95, p. 102026, 2022. doi:10.1016/j.compmedimag.2021.102026

[8] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks", *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009. doi:10.1016/j.ipm.2009.03.002