Softplus Meta-Modeling for Chick Counting: Extending YOLO-Based Coefficient Models

Logan Kelsch (Chick Counting Group)
COSC 426 (Software Engineering II)

October 20, 2025

Summary

In my previous documented analysis, on MCPT, we validated a very simple one-parameter coefficient meta-model on top of a strong RGB YOLO detector for chick counting. The YOLO detector is excellent at finding chick-like blobs, but it systematically undercounts the total number of chicks because it does not reliably distinguish single chicks from tightly packed groups. That undercount bias was shown to be highly consistent, which made a scalar coefficient an effective first-order correction. With statistically defended the approach with a Monte Carlo Permutation Test.

In this document we extend that correction idea by replacing the scalar coefficient with a lightweight linear meta-model whose output is passed through a softplus nonlinearity. The goal is to keep the meta-model simple enough to remain interpretable and statistically analyzable, while giving it enough flexibility to correct not just a fixed bias but also small systematic effects that depend on the structure of the YOLO predictions (for example, how many overlapping bounding boxes appear, or how packed chicks are in frame).

Softplus Linear Meta-Model on YOLO Detections

At a high level, we treat the YOLO detector as a feature generator rather than a final counter. For each capture (bounding box crossing a detection line) we compute a feature vector $x \in \mathbb{R}^d$ derived from the RGB data of the bounding box. Examples of such features include:

- Bounding box size and proportion.
- Analysis of yellow contents in the box.
- Simple analysis of colors in the box.

The meta-model is a linear function of these features, followed by a softplus to ensure non-negative predicted counts. The model is trained with stochastic gradient descent (SGD) on a per-capture loss (Anye's hand count values at various frames)

The softplus function is a smooth, non-negative alternative to the ReLU:

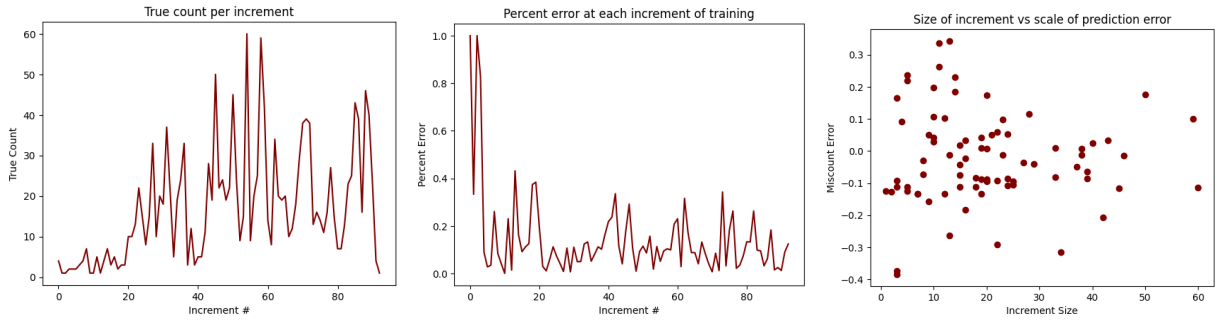
$$\text{softplus}(z) = \log\big(1 + e^z\big).$$

Intuitively, for large positive z it behaves like z, and for large negative z it smoothly decays toward 0. This makes it well-suited for modeling non-negative quantities like counts or rates while still being friendly to gradient-based optimization.

To keep the evaluation aligned with how the system will be used operationally, we adopt a walk-forward scheme. Captures are ordered as they would arrive in practice, and at each step we update the parameters using the past data and then evaluate on the next increment of true count (these increment sizes vary largely with most from 10-50). This gives a natural view of how the meta-model improves and then plateaus over time as more labeled data is seen.

Below we leave space for three illustrative plots such as:

- True Count size of increments across time,
- walk-forward accuracy across increments,
- Scatter plot of scaled error vs. increment size.



Coupling IoU Thresholds with Linear-Model Learning Rate

The YOLO detector has several tunable hyperparameters, and the Intersection-over-Union (IoU) threshold is one of the most influential. Lowering the IoU threshold typically reduces the number of overlapping detections, which may increase overcounting but can also raise marginal detections in crowded scenes. At the same time, the learning rate (LR) of the SGD optimization in the softplus meta-model controls how quickly the linear layer adapts to the error structure induced by each IoU choice.

Rather than treat these two pieces independently, we can jointly explore IoU threshold and learning rate as a (IoU, LR) grid, as these walk-forward models can be generated and evaluated in short time:

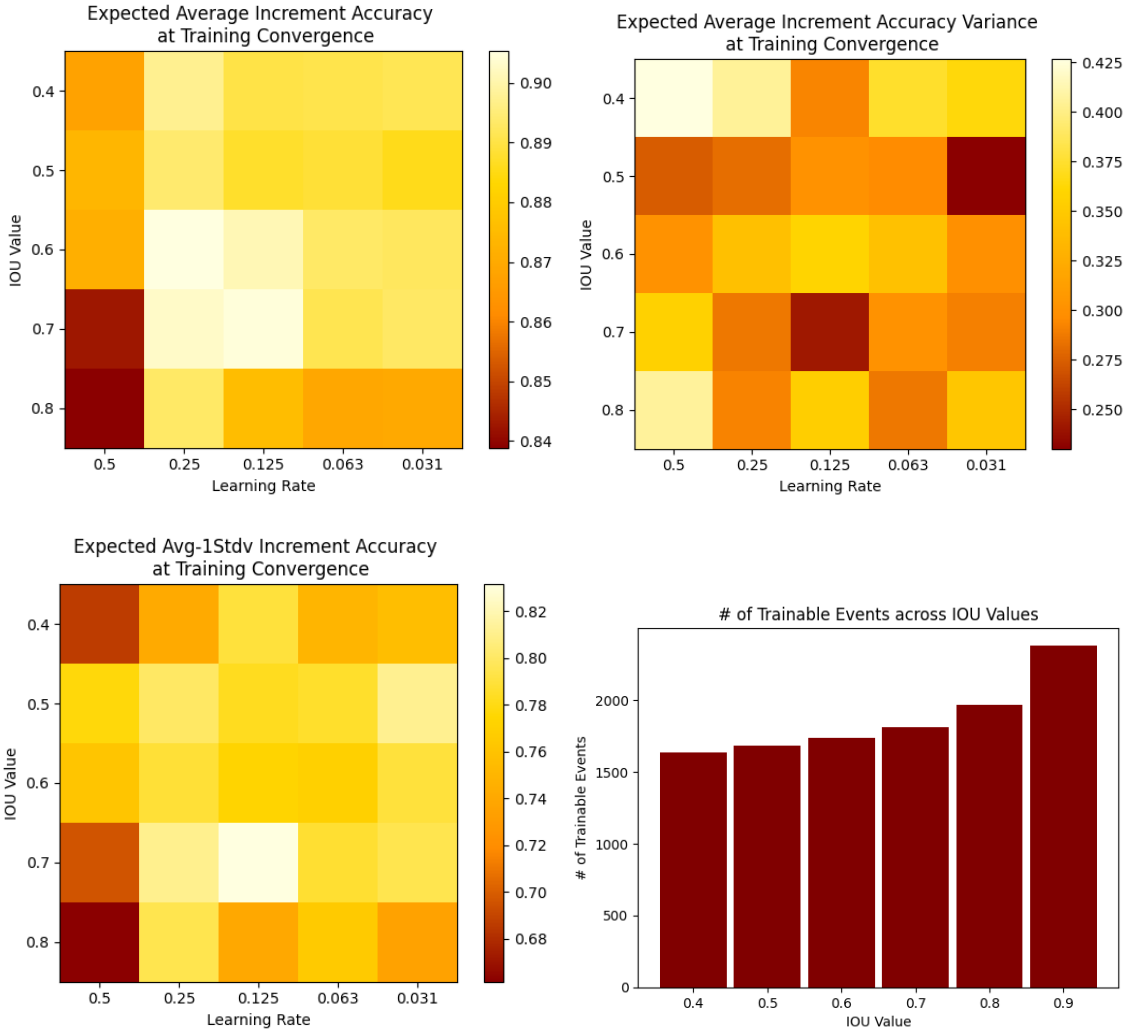
1. Choose a set of IoU thresholds, e.g. $t_{\text{IoU}} \in \{0.5, 0.6, \ldots, 0.8\}$.
2. Choose a set of learning rates, e.g. $\eta \in \{2^{-1}, 2^{-2}, \ldots, 2^{-5}\}$.
3. For each (t_{IoU}, η) pair, run a walk-forward training procedure of the softplus meta-model on YOLO-derived features and (after plateau) record:
   - the average increment error,
   - the variance of magnitude of increment error,
   - number of trainable events (static across η dimension).

Visualizing this grid helps answer questions like:

- Are there IoU regimes where the meta-model is easy to train (wide range of stable learning rates)?
- Are there IoU choices that induce high variance errors that the linear layer cannot absorb at any reasonable learning rate?

Below we leave space for four images:

- average increment accuracy over (IoU, LR),
- standard deviation of magnitude over (IoU, LR),
- average increment accuracy under -1 magnitude standard deviation over (IoU, LR),
- Visualization of number of trainable events across IoU.



Exponential Fit to Walk-Forward Performance and Post-Plateau Error

When we look at walk-forward performance over many increments (for example, over hundreds or thousands of additional chicks), a common pattern emerges:

1. Early increments show rapid improvement as the meta-model parameters adapt to the data.
2. After some point, performance plateaus: the cumulative percent error fluctuates around a stable mean with a relatively tight band.

A simple way to summarize this behavior is to approximate the performance metric (e.g. increment error) as an exponential approach to a plateau:

$$g(t) = \alpha\, e^{-\beta t} + \gamma,$$

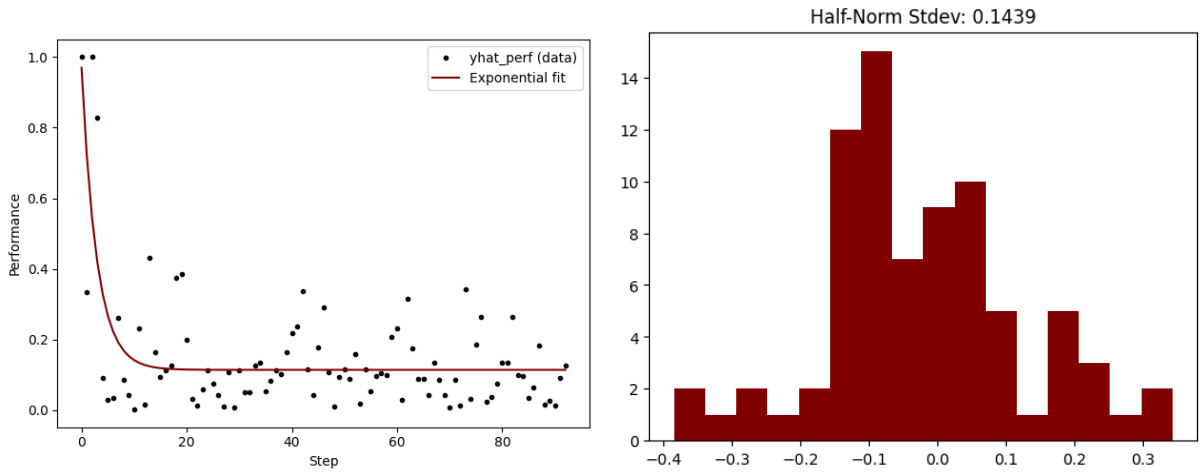
where t indexes the number of walk-forward increments, α controls the initial gap from the plateau, β is a decay rate, and γ is the asymptotic error level (the plateau). Fitting this curve to the observed walk-forward trajectory gives:

- a compact summary of how quickly the meta-model converges, and
- a stable estimate of long-run average error.

Once we identify a plateau region (for example, all increments after a certain t_{plateau}), we can also examine the distribution of residual errors in that region. A simple diagnostic is a histogram of per-increment errors after the plateau, which helps confirm whether the remaining noise is roughly symmetric and small, or whether there are systematic tails we still need to address.

Below we leave space for two images, for example:

- the walk-forward performance curve with an exponential fit overlaid,
- a histogram of increment errors in the plateau region.



Diminishing Error Envelope over Repeated Events

A natural question for operational decision-making is: how does error accumulate (or cancel) as we aggregate more and more increments? Even if the per-increment error can be a few percent, random over- and under-counts tend to cancel out over many independent increments. This leads to a diminishing-error effect similar to the familiar $1/\sqrt{t}$ behavior from the central limit theorem.

Suppose for the moment that the per-increment error e_i is approximately zero-mean with variance σ^2 and is independent across increments. Consider the average error after t increments,

$$\bar{e}_t = \frac{1}{t}\sum_{i=1}^{t} e_i.$$

Then, under a Gaussian approximation, $\bar{e}_t$ has variance σ^2/t, and the expected absolute value of the average error satisfies

$$\mathbb{E}\big[\,|\bar{e}_t|\,\big] \;\approx\; \sigma\sqrt{\frac{2}{\pi t}}.$$

This expression gives a theoretical envelope for how the *typical* magnitude of average error should shrink as we aggregate more increments.

In practice, our increments are heteroscedastic: smaller true counts tend to produce higher varying errors. We can encode this by letting the standard deviation of the *relative* error depend on the true size y of the increment, for example through a function $\sigma_r(y)$. Then the cumulative percent error after t increments can be studied as:

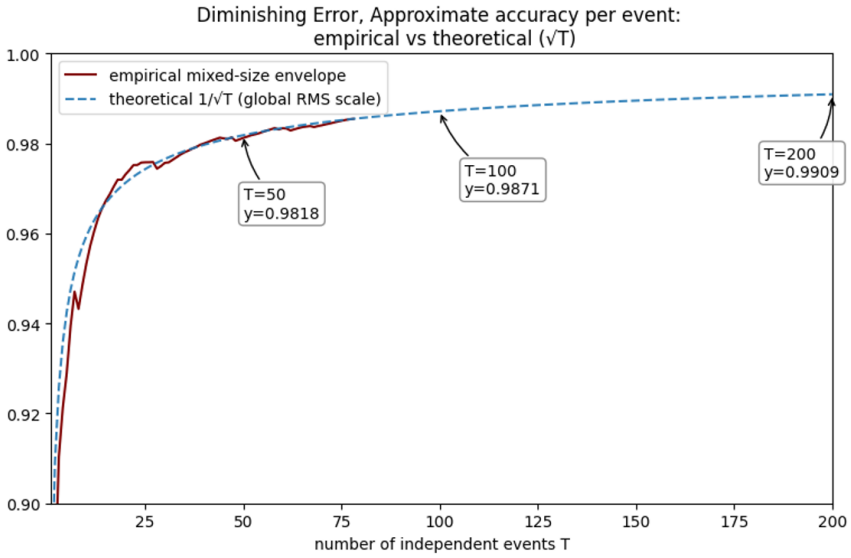
$$\mathrm{CPE}_t = \frac{\sum_{i=1}^{t} y_i r_i}{\sum_{i=1}^{t} y_i},$$

where r_i is the relative error on increment i. When increments are reasonably balanced and r_i are independent, the same kind of $1/\sqrt{t}$ diminishing envelope applies, but now the effective σ is a function of the distribution of y_i and $\sigma_r(y_i)$.

Operationally, we can:

1. Compute empirical CPE trajectories for aggregated blocks of size t (as we want to estimate for 100 chicks).
2. Overlay the theoretical envelope $\sigma_{\text{eff}}\sqrt{2/(\pi t)}$ fitted from the heteroscedastic residuals.
3. Fit a smooth curve to these envelopes to quickly estimate the typical CPE at $t = 50$, $t = 100$, and $t = 200$ chicks.

The image beloe is a plot of empirical CPE envelope vs. t, with the theoretical envelope representation (exponential fit), and markers at $t = 50$, 100, and 200.



Monte Carlo Permutation Test for Cumulative Percent Error

To go beyond analytic approximations, we can use a Monte Carlo permutation test (MCPT) to explore the range of possible cumulative percent errors over realistic random walks of future increments.

First, we formalize a simple heteroscedastic error model at the increment level:

$$r \mid y \sim \mathcal{N}\big(0, \sigma_r^2(y)\big),$$

where r is the relative error (e.g. fractional deviation from the true count) on an increment of true size y and $\sigma_r(y)$ is estimated from the meta-model residuals as a function of y.

For a walk of T increments with true sizes $y_1, \ldots, y_T$ and relative errors $r_1, \ldots, r_T$, the cumulative percent error is defined as

$$\mathrm{CPE} = \frac{\sum_{i=1}^{T} y_i r_i}{\sum_{i=1}^{T} y_i}.$$

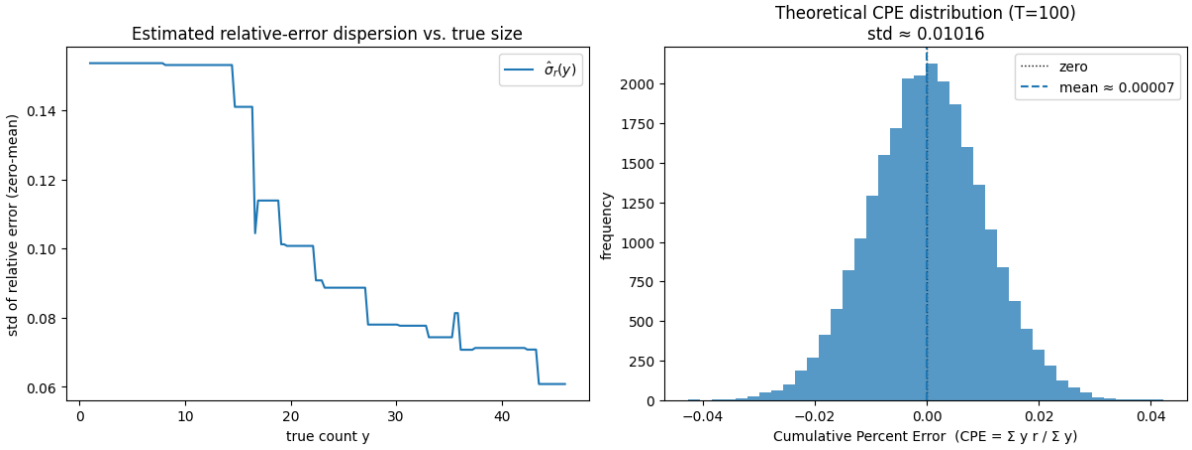
In the MCPT, we proceed as follows:

1. Estimate $\sigma_r(y)$ from the observed residuals of the softplus meta-model (for example, via a smooth regression of $|r|$ or r^2 against y).
2. Generate many synthetic walks (25,000) of length $T = 100$:
   - For each walk, either sample (y_i) from the empirical distribution of observed increments, or permute an existing sequence.
   - For each y_i, draw $r_i \sim \mathcal{N}(0, \sigma_r^2(y_i))$ and compute the corresponding CPE.
3. Aggregate the resulting CPE values across simulations to obtain:
   - an empirical distribution of CPE for $T = 100$,
   - empirical standard deviation to show probabilities of better or worse performance.

This Monte Carlo view complements the analytic $1/\sqrt{t}$ envelope: instead of assuming homogeneous increments, it respects the actual mix of true sizes and the heteroscedastic residual structure learned from the limited data.

Below we leave space for an image that combines two views:

- a curve of estimated relative-error dispersion $\sigma_r(y)$ vs. true size y, and
- the MCPT distribution of CPE for $T = 100$ (and mean and deviation values).



Conclusion

Starting from a simple coefficient correction on top of YOLO detections, we have outlined a more expressive but still lightweight meta-model for chick counting. By framing the YOLO output as a feature generator and applying a softplus-transformed linear layer trained via SGD, we maintain interpretability and statistical tractability while allowing the meta-model to absorb structured biases and mild nonlinearities.

These results conclude that using carefully placed camera's data paired with machine learning, we can generate an approach to counting 100 chicks with an estimated accuracy of at least $98.7 \pm 1.0\%$, which is a reduction of error of over 55%, with an unknown, but large, reduction in error variance. Thus, employment of this approach or consideration of further validation or development is obvious given the potential reduction of cost, and given the passion in the heart of the programmer writing this.

For questions about these methods or figures, please contact:
Logan Kelsch
`lkelsch1@gulls.salisbury.edu`

References

[1] Dwass, M. (1957). Modified Randomization Tests for Nonparametric Hypotheses. *Annals of Mathematical Statistics*, 28(1), 181–187.

[2] Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. In *Proceedings of NeurIPS 2017*, 4765–4774.

[3] Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of COMPSTAT 2010*, 177–186.

[4] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.