Expanding the GLM Toolkit for Transformers as Statisticians

Joseph Tennyson
University of California, Berkeley
josephtennyson@berkeley.edu

Logan King
University of California, Berkeley
loganking@berkeley.edu

Derry Xu
University of California, Berkeley
derryxu@berkeley.edu

Connor Chen
University of California, Berkeley
connorchen@berkeley.edu

Abstract

This study extends prior research [1] on the capability of transformer models to approximate generalized linear models (GLMs) through in-context learning. While previous work primarily focused on binary and continuous response variables, we broaden the scope by incorporating additional GLM classes, specifically poisson and negative binomial regression models. Furthermore, we explore the transformer's ability to differentiate among various GLMs based solely on in-context data. We aim to demonstrate that transformers can not only discern among broadly different types of data (ex. binary vs continuous) as done in previous work, but can also perform nuanced algorithm selection tasks within a single data type, exemplified by distinguishing between poisson and negative binomial regression models based on observed over-dispersion in count data. This capability further supports the concept of the "transformer-as-a-statistician", [1], underscoring the potential of transformers to autonomously perform sophisticated statistical analyses. Ultimately, we report mixed results, demonstrating success in joint training with some limitations.

1 Introduction

Generalized linear models are a foundational class of statistical models, encompassing a wide range of common learning algorithms including but not limited to linear regression, logistic regression, and poisson regression.

GLMs presume that the response variables of interest are random, and distributed according to some common distribution. Based on the GLM model, the expectation of the response variables conditioned on the observed covariates is modeled by some potentially non-linear transformation of a linear combination of covariates. The goal of a GLM is to approximate this unknown weight vector using maximum likelihood estimation.

The utility of generalized linear models is apparent: while maintaining the statistical convenience of linear regression techniques, GLMs are able to model different types of data, determined by the support of the distribution of the response. These convenient features have made GLMs an essential tool in statistical modeling.

Recent work has shown that transformer based architectures can be trained such that they gain in-context learning (ICL) abilities, where the transformer is able to perform novel tasks when provided with a few in-context examples. Transformers have been shown to have the ability to learn several different function classes in context [2], including certain GLMs (e.g. linear regression and logistic

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

regression) [1]. In essence, a transformer model at inference time on in-context examples can approximate the process by which GLMs fit to training data.

Further, studies have shown that suitably trained transformers have model-selection capabilities [1]: given the context, the model can identify whether it is a classification or regression problem, and nearly match the performance of linear regression on regression problems and logistic regression on classification problem with a single trained GPT-2 model.

We aim to expand on the work done in "Can Transformers Learn Sequential Function Classes In Context?" [1] by applying a similar recipe to a wider variety of GLMs. We first train several individual models that can emulate the capabilities of different types of GLMs in context. Specifically, we learn poisson and negative binomial regressions, both of which model count data.

We then investigate the capability of a GPT-2 style transformer to conduct more nuanced and advanced model selection tasks. When modeling count data, a poisson regression is preferable when the response variables are not over-dispersed, i.e. the expectation of the response is approximately similar to the variance of the response. When the data is over-dispersed, a negative binomial regression is preferable. Thus, we seek to train a transformer model which can learn and apply poisson and negative binomial regression in the appropriate contexts.

We find that joint training is effective in specific scenarios, specifically when the jointly learned distributions are not overly different. However, when the distributions are significantly different, joint performance suffers.

2 Related Works

There has been substantial work already done in exploring the capabilities of "transformers as statisticians."

In [2], the authors first propose exploring in-context capabilities of LLMs (Large Language Models) by studying the more restricted problem of training a transformer to in-context learn to approximate different simple functions. One of the simple functions learned by an transformer model was a noised linear function, essentially learning a linear regression.

[1] expands on the previous work by further investigating the capabilities of transformers to in-context learn various statistical models, which include linear and logistic regression, but also Lasso and Ridge regularized linear regressions. They also explored the model-selection capabilities of transformers by training a model to both conduct in-context classification and regression tasks, and also by training a model to select an appropriate regularization hyperparameter given different contexts. Finally, the concept of "transformers as statisticians" which we adopt was first coined in this paper. Our paper can be seen as an extension of the joint regression and classification model introduced in this paper.

[3] also explores the model selection capabilities of transformers through an empirical risk minimization perspective, and show that transformers can in-context learn optimal ridge/weighted solutions, including on autoregressive linear models using a sliding window approach. While the paper mainly focuses on parameter selection for ridge regression tasks, the core idea of using transformers to automatically conduct model selection is shared with our work.

[4] extends in-context learning by testing transformers on complex learning tasks beyond linear settings. This includes examining whether transformers can use learned behaviors and apply them in novel tasks. however, this work primarily uses low-degree polynomial functions, and does not look into learning higher-degree polynomials.

[5] explores using transformers for in-context learning in statistical inference tasks, more specifically parameter estimation in a Bayesian setting. They show that a transformer can match the performance of traditional Empirical Bayes methods in learning a poisson distribution.

Our paper primarily extends on the work of [1] and [3]. Rather than only considering model selection between continuous regression and binary classification problems, we expand the possible set of problems to include count data. Further, we demonstrate that transformers' model selection capabilities go beyond hyperparameter searching, and can also effectively select the family of models most appropriate to minimize loss.

3 Background

3.1 Generalized Linear Models

Given a feature matrix $X \in \mathbb{R}^{n \times d}$ and a response vector $Y \in \mathbb{R}^n$, a generalized linear model is composed of three main components:

- A distribution that models response variable Y
- A linear combination of covariates $X\beta$, where $\beta \in \mathbb{R}^d$
- A function $g^{-1}(\cdot)$, such that $\mathbb{E}[Y \mid X] = g^{-1}(X\beta)$, dubbed the inverse-link function.

GLMs preserve a linear component, but can be non-linear functions depending on the designated inverse-link function.

The weights are typically fit through maximum likelihood estimation methods.

An advantage of GLMs is that each GLM is associated with a sampling distribution for the responses. The GLM assumes that our observed Y values are sampled from this distribution. When this assumption is true, the GLM is asymptotically the Bayes-optimal predictor under the implied loss function from the distribution of Y.

This means that when given some covariates X and some true weights β, if we generate our observed responses Y according to the data-generating process implied by the GLM we are trying to in-context learn, then the GLM is optimal for estimating w and can serve as an upper bound on the performance of the GPT-2 style model.

Below we list out the relevant GLM recipes to our investigation.

3.1.1 Poisson Regression

$$
\begin{aligned}
&Y \sim \text{Poisson}(\exp(X\beta)) \\
&E[Y] = \exp(X\beta)
\end{aligned}
$$

The inverse-link function is the exponential function $f(x) = \exp(x)$, and the distribution of the response is Poisson.

This is appropriate for modeling count data, since the support of the response distribution is $\mathbb{N}_0$, or the natural numbers starting from 0.

It is important to note that this model assumes that $\text{Var}[Y] = \exp(X\beta)$. This means that if the true response has a variance greater than $\exp(X\beta)$, the poisson model is inaccurate. This problem is called overdispersion, and can lead to an inaccurate fit and predictions.

3.1.2 Negative Binomial Regression

$$
\begin{aligned}
&\mu = \exp(X\beta) \\
&Y \sim \text{NegBin}(\mu, \mu + \alpha\mu^2) \\
&E[Y] = \mu \\
&\text{Var}[Y] = \mu + \alpha\mu^2
\end{aligned}
$$

The inverse-link function is the exponential function $f(x) = \exp(x)$, and the distribution of the response is negative binomial.

Note that the negative binomial parameterization used above is the mean-variance parameterization, often called NB2. To convert back to the typical parameterization using r, the number of successes to be observed, and p the probability of success, we can use the following equations:

$$
\begin{aligned}
r &= \frac{1}{\alpha} \\
p &= \frac{1}{1 + \alpha\mu}
\end{aligned}
$$

This model is also appropriate for modeling count data, since the support of the response distribution is $\mathbb{N}_0$, or the natural numbers starting from 0.

Unlike the poisson model, the negative binomial model allows for a learned variance controlled by α. During the typical learning algorithm for negative binomial models, α is also learned, accounting for overdispersion. The negative binomial model can be seen as a generalization of the poisson model; when $\alpha \to 0$, the two regressions are the same.

4 Training Setup

We train a GPT-2-style Transformer denoted M_θ to perform in-context learning of arbitrary generalized linear models. We provide model details, prompt construction, training procedure, and validation strategy.

4.1 Model Details

- **Model Architecture:** 12 layers, 8 attention heads per layer, hidden dimension of size 256.
- **Optimization:** Adam with $\mathrm{lr} = 2.5 \times 10^{-4}$ (unless otherwise specified), batch size 256, no curriculum. Each batch samples 256 fresh GLM tasks.
- **Training Schedule:** 8,000 gradient steps (unless otherwise specified), with model checkpoints every 1,000 steps.
- **Input Dimensionality:** $d = 10$ by default.

4.2 In-context Prompt Construction

For each training example (a "task"), we simulate a GLM with exponential link:

1. Sample a true weight vector:
$$w \ \sim \ \mathcal{N}(0, \sigma^2 I_d) \quad \text{(gaussian prior with variance } \sigma^2 \text{ to control label variance)}.$$
2. Sample 40 feature vectors:
$$x_i \ \sim \ \mathcal{N}(0, I_d), \quad i = 1, \ldots, 40.$$
3. Compute the mean parameter for each sample using the GLM link function:
$$\mu_i \ = \ \exp(\text{clip}(w^\top x_i, \ -4, \ 4)).$$
Then sample targets y_i from the GLM distribution with mean μ_i; for example,
$$y_i \ \sim \ \text{Poisson}(\mu_i)$$
4. We create 40 in-context pairs (x_i, y_i) for $i = 1, \ldots, 40$.

Why utilize σ: Each component $w_j x_{ij}$ is a product of independent gaussians with variance σ^2, so the variance of $w^\top x_i = \sum_j w_j x_{ij}$ is $\sigma^2 \|x_i\|^2$. Since the exponential function grows rapidly, high variance in $w^\top x_i$ can cause extreme values in y_i, destabilizing training. To mitigate this, we clip $w^\top x_i$ to the range $[-4, 4]$ before applying $\exp(\cdot)$, which bounds the maximum value of y_i. Reducing σ further concentrates $w^\top x_i$ near zero, leading to more stable targets and improved training behavior. For our experiments, we utilize $\sigma^2 = 0.32$.

4.3 Loss Function

At training time, the model predicts each label y_i using the current input x_i and all previous in-context pairs $\{(x_j, y_j)\}_{j<i}$. Formally, the predicted natural parameter is

$$\hat{\eta}_i = M_\theta\big(P_{<i}, x_i\big),$$

where $P_{<i} = \{(x_j, y_j)\}_{j<i}$ is the in-context prefix. The loss incurred at each step is the exponential-family negative log-likelihood:

$$\ell_{\mathrm{NLL}}(\hat{\eta}_i; y_i) = A(\hat{\eta}_i) \ - \ y_i \, \hat{\eta}_i \ + \ C(y_i).$$

Function specific forms of this loss are described in the experiments section.

4.4 Validation and Metrics

After training, we evaluate on a held-out set of 10,000 fresh tasks (sampled as above). For each task and for each predictor (model, true-weight oracle, MAP oracle) we compute

$$\mathrm{NLL} \;=\; \frac{1}{40}\sum_{i=1}^{40} \ell\big(\hat{\eta}_i; y_i\big) \quad \text{where} \quad \ell(\hat{\eta}; y) = A(\hat{\eta}) - y\,\hat{\eta} + C(y).$$

We then report the average NLL over all tasks for:

- The trained model M_θ.
- True-weight (Bayes) oracle.
- Gradient-descent (MAP) oracle.

4.5 Oracle predictors and Optimality Proofs

We compare M_θ against two idealized predictors.

4.5.1 True-Weight (Bayes) Oracle

This predictor knows the true parameter w and for any input x outputs

$$\hat{\eta}_{\mathrm{TW}}(x) = w^\top x, \qquad \hat{y}_{\mathrm{TW}} = g^{-1}\big(w^\top x\big).$$

Optimality
Under a GLM with natural-parameter link and log-partition function A, the per-example loss is

$$\ell(\hat{\eta}; y) = A(\hat{\eta}) \;-\; y\,\hat{\eta} \;+\; C(y).$$

The expected NLL at input x is

$$\mathbb{E}_{Y|x}\big[\ell(\hat{\eta}; Y)\big] = A(\hat{\eta}) - \mathbb{E}[Y \mid x]\,\hat{\eta} + \text{const.} = A(\hat{\eta}) - A'\big(w^\top x\big)\,\hat{\eta} + \text{const.}$$

Differentiating w.r.t. $\hat{\eta}$ and setting to zero gives

$$A'(\hat{\eta}) \;-\; A'\big(w^\top x\big) = 0 \quad \Longrightarrow \quad \hat{\eta} = w^\top x,$$

so the true-weight oracle minimizes the expected NLL.

4.5.2 Gradient-Descent (MAP) Oracle

We place a gaussian prior $w \sim \mathcal{N}(0, \sigma^2 I)$ and use the same GLM loss above. Given in-context data $\{(x_i, y_i)\}_{i=1}^{40}$, the MAP estimate is

$$\hat{w}_{\mathrm{MAP}} = \arg\min_w \left\{ \sum_{i=1}^{40} \ell\big(w^\top x_i; y_i\big) \;+\; \frac{1}{2\sigma^2}\|w\|^2 \right\},$$

and its predictions on a new x are

$$\hat{\eta}_{\mathrm{MAP}}(x) = \hat{w}_{\mathrm{MAP}}^\top x, \qquad \hat{y}_{\mathrm{MAP}} = g^{-1}\big(\hat{w}_{\mathrm{MAP}}^\top x\big).$$

Optimality of the MAP Oracle
The negative logarithmic posterior (up to additive constants) is

$$L(w) = \sum_{i=1}^{40} \ell\big(w^\top x_i; y_i\big) \;+\; \frac{1}{2\sigma^2}\|w\|^2.$$

Since each $\ell(\cdot\,; y_i)$ is convex in w (because A is convex) and the quadratic prior term is strictly convex, $L(w)$ is strictly convex. Hence there is a unique minimizer $\hat{w}_{\mathrm{MAP}}$, characterized by

$$\nabla_w L(w) \;=\; \sum_{i=1}^{40} \big[A'\big(w^\top x_i\big) - y_i\big]\, x_i \;+\; \frac{1}{\sigma^2} w = 0.$$

Standard gradient descent converges to this global optimum, which therefore minimizes the regularized NLL.

5 Individual Training

5.1 Poisson Training

The first novel function class we consider is the poisson generalized linear model, which is commonly used for modeling count data. In this setting, each label $y \in \mathbb{N}_0$ is assumed to follow a poisson distribution with mean, $\mu = \exp(w^T x)$ where $x \in \mathbb{R}^d$ is the input vector and $w \in \mathbb{R}$ is the weight vector. The conditional distribution is given by:

$$P(y|x; w) = \frac{e^{yw^T x - \exp(w^T x)}}{y!} \tag{1}$$

To train the transformer to in-context learn this function class, we utilize the poisson negative log-likelihood loss, implemented via PyTorch's `torch.nn.PoissonNLLLoss`. This corresponds to the full loss

$$\mathcal{L}_{Poisson}(w; x, y) = \exp(w^T x) - yw^T x + log(y!) \tag{2}$$

which penalizes deviations from the expected poisson count while properly accounting for the variance and distributional structure of the target variable. This setup allows the model to learn not only the expected value of the counts but also the correct generative structure associated with poisson distributed data.

We observe in Figure 1 that our model is able to match the performance of gradient descent optimization. This result demonstrates that in-context learning is capable of approximating the solution quality of a fully optimized model.

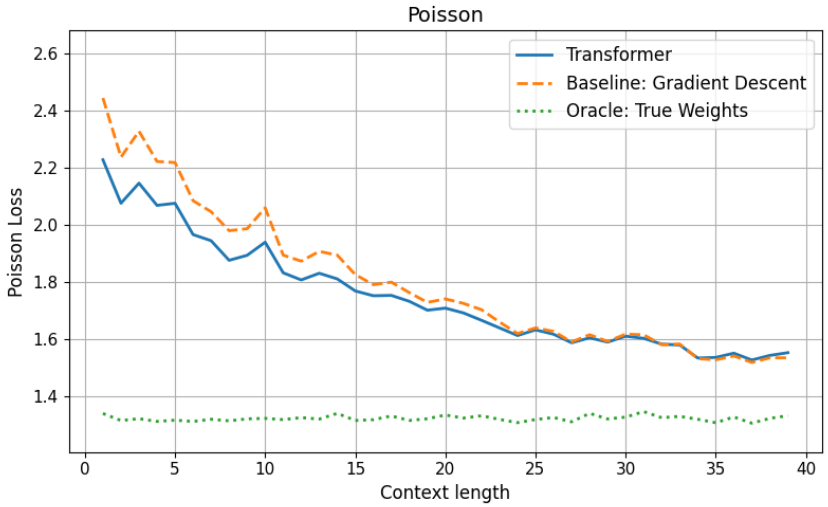


Figure 1: **In-context learning of Poisson Regression.** We compare our model to the gradient descent and true weights oracle using poison negative log likelihood loss. Our model achieves performance similar to gradient descent on this validation set of 10,000 points.

5.2 Negative Binomial Training

The next function class we examine is the negative binomial GLM, which generalizes poisson regression by allowing for over-dispersed count data; situations where the variance of the response exceeds its mean.

In this setting, the expected count is modeled as:

$$\mu = \exp(w^\top x),$$

and the response variable $Y \in \mathbb{N}_0$ follows a negative binomial distribution with fixed dispersion parameter $r > 0$:

$$Y \sim \text{NegBin}(r, \mu).$$

This corresponds to a formulation where the variance is given by:

$$\text{Var}[Y] = \mu + \frac{\mu^2}{r}.$$

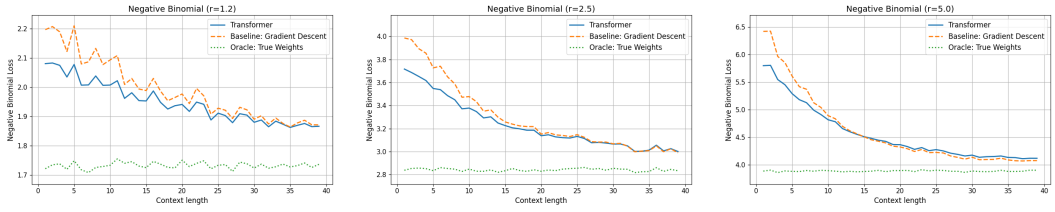
Unlike the poisson model, which assumes $\text{Var}[Y] = \mu$, this setup allows for controlled overdispersion via the hyperparameter r, which we fix at training and evaluation time. When $r \to \infty$, the negative binomial model approaches the poisson distribution, but with small r, the variance becomes significantly larger than the mean.

We study the effect of varying overdispersion by evaluating three fixed values of the dispersion parameter $r \in \{1.2,\ 2.5,\ 5.0\}$ in isolation. We use the negative log-likelihood loss for the negative binomial distribution with fixed r. We sample from a negative binomial distribution implemented via PyTorch's `torch.distributions.NegativeBinomial` and define a loss. This yields:

$$\mathcal{L}_{\text{NB}}(y, \mu) = -\log\left[\binom{y + r - 1}{y}\left(\frac{r}{r+\mu}\right)^r\left(\frac{\mu}{r+\mu}\right)^y\right],$$

where $y \in \mathbb{N}_0$ is the observed count, the distribution is parameterized by the mean μ and dispersion r, and the binomial coefficient is defined via the gamma function as:

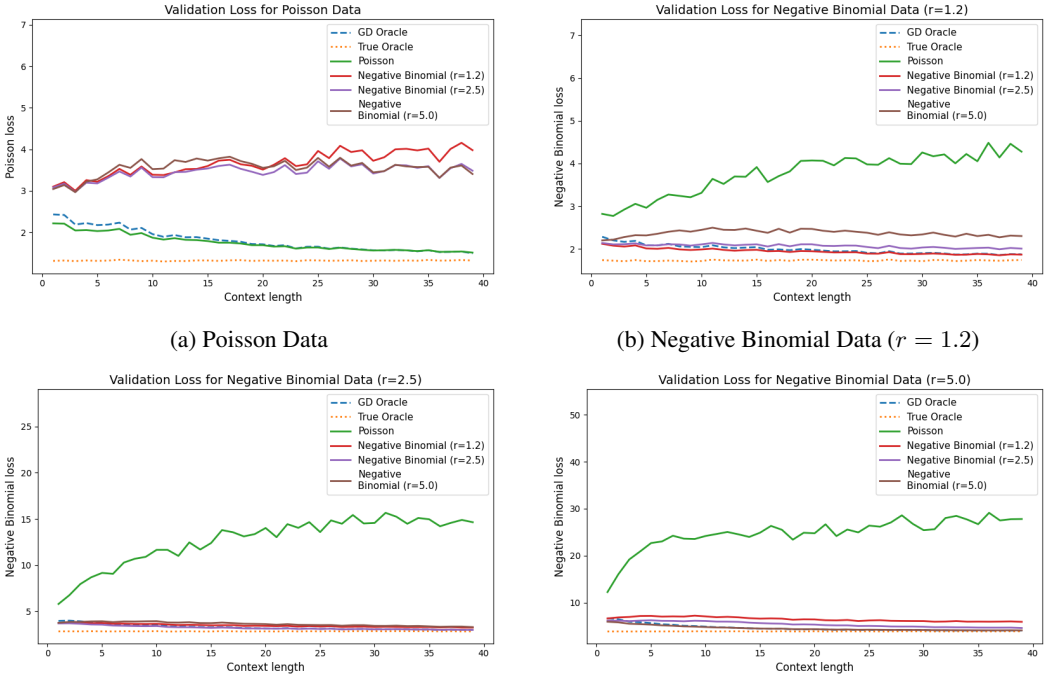
$$\binom{y + r - 1}{y} = \frac{\Gamma(y + r)}{\Gamma(r)\,\Gamma(y + 1)}.$$



(a) Negative Binomial with $r = 1.2$ (b) Negative Binomial with $r = 2.5$ (c) Negative Binomial with $r = 5.0$

Figure 2: **In-Context Learning of Negative Binomial Regression.** We evaluate model performance at varying dispersion values $r \in \{1.2, 2.5, 5.0\}$ to assess whether different negative binomial distributions can be learned. All three formulations converge to gradient descent performance.

Our results in 2 show that all 3 version of negative binomial regression are learned at the level of gradient descent. Furthermore, we investigate generalization by investigating the performance of the poisson model as well as all negative binomial models on all data types. In figure 3, we see that negative binomial models generalize well to each other while the poisson model does not generalize.



(a) Poisson Data

(b) Negative Binomial Data ($r = 1.2$)

(c) Negative Binomial Data ($r = 2.5$)

(d) Negative Binomial Data ($r = 5.0$)

Figure 3: **Validation Performance across Distribution Families.** Each plot shows the validation loss for different models on a held-out dataset, across poisson and various negative binomial regression settings. A common trend emerges where models trained with negative binomial data generalize fairly well to negative binomial distributions with different r values. However, models trained with poisson data do poorly on non-poisson tasks.

6 Joint Training

In the joint training setting, our model is trained to perform in-context learning across multiple GLM families. Specifically, we train the model on a mixture of poisson and negative binomial regression tasks, without revealing the task type. The goal is for the model to implicitly infer the appropriate statistical model from the in-context examples and adapt its prediction strategy accordingly.

Poisson and negative binomial regression both model count data, but differ significantly in how they model variance. While poisson assumes that the variance equals the mean, the negative binomial allows for overdispersion via a fixed dispersion parameter r. This setup presents a more nuanced model selection challenge: the two distributions can yield similar samples when dispersion is low, yet diverge meaningfully in their variance structure. Successful joint training requires the transformer to detect subtle statistical cues such as increasing variance with respect to the mean and should select the appropriate inference strategy.

At each training step, the model receives a batch of in-context learning prompts, where each prompt is randomly sampled from one of the two distributions with probability $\frac{1}{2}$:

a. **Poisson regression task**: $y_i \sim \text{Poisson}(\exp(w^\top x_i))$

b. **Negative Binomial regression task**: $y_i \sim \text{NegBin}(r, \mu = \exp(w^\top x_i))$,

where $w \sim \mathcal{N}(0, \sigma_d^2)$, $x_i \sim \mathcal{N}(0, I_d)$.

We jointly train three separate models, each mixing poisson data with a specific negative binomial distribution ($r \in \{1.2, 2.5, 5.0\}$).

The model is not given any task identifier. Instead, it must infer the appropriate predictive strategy based purely on the statistics of the in-context examples. The loss function used for each prompt corresponds to the true underlying distribution:

$$\mathcal{L}_{\text{joint}}(y, \hat{y}) = \mathcal{L}_{\text{Poisson}}(y, \hat{y}) \cdot \mathbf{1}_{\{\text{Poisson}\}} + \mathcal{L}_{\text{NB}}(y, \hat{y}) \cdot \mathbf{1}_{\{\text{NB}\}}$$

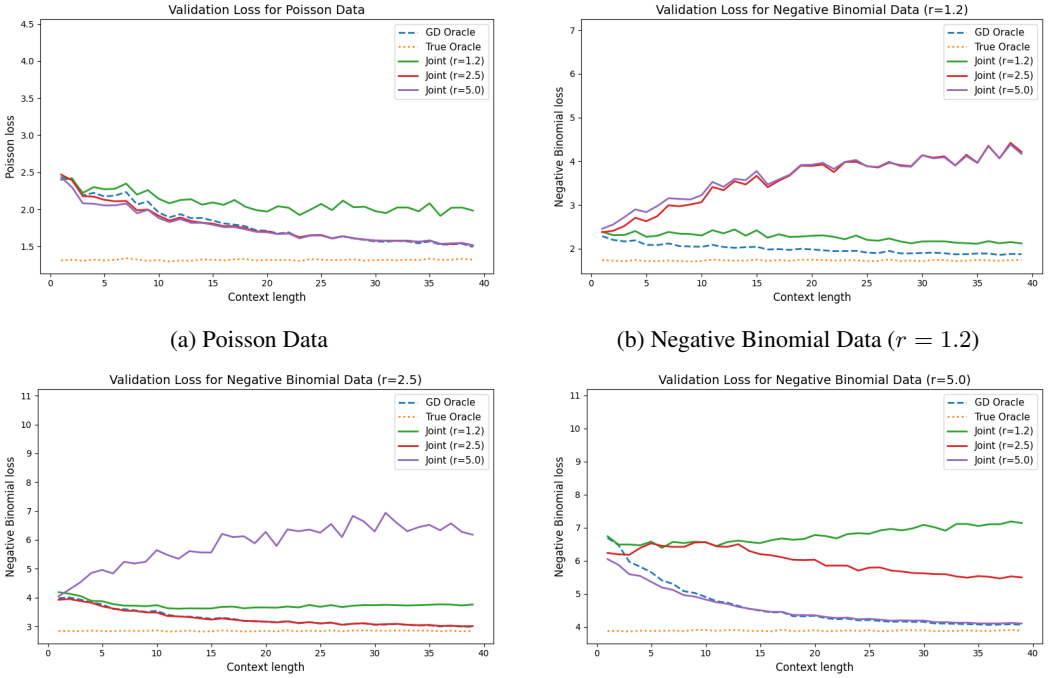
For $r = 2.5$ and $r = 5.0$, the learned model closely tracks the gradient descent oracle in 4. However, for $r = 1.2$, the model struggles to match oracle performance.

Furthermore, we notice that loss is generally higher for negative binomial distributions with larger r values in 4, though convergence to the oracle is better in those cases. Referencing the training plots for the r = 1.2 case in the appendix (7, we see that the model learns the poisson distribution much better than the negative binomial one. This suggests a trade-off induced by the joint training objective. When r is small, the NB distribution is highly overdispersed, and the corresponding NB loss landscape is smoother and less sensitive to small prediction errors. This results in smaller gradients from the NB component. In joint training, these weaker gradients mean the NB objective has less influence on the overall parameter updates. This allows the optimizer to focus more effectively on objectives with sharper loss landscapes or stronger gradients (like the poisson component, assuming its landscape is relatively sharper), potentially leading to better performance on those tasks but struggling to significantly reduce the NB loss where the optimization signal is weaker. Conversely, when r is large, the NB distribution becomes more poisson-like (less overdispersed), and the NB loss landscape becomes sharper and more sensitive to errors, generating larger gradients. This gives the NB objective more influence in the joint updates.

This trade-off highlights a central tension in joint training across heterogeneous objectives. The dispersion parameter r not only affects the statistical properties of the distribution but also implicitly controls the relative influence or importance of the NB task in the loss landscape through the magnitude of its gradients. A smaller r reduces the gradient magnitude of the NB loss, diminishing its influence during gradient-based updates relative to other tasks. This can lead to better performance on tasks with more strongly shaped loss surfaces (e.g., poisson), potentially at the expense of underfitting the NB task which provides a weaker optimization signal.

These findings suggest that careful tuning of per task weighting or adaptive scaling strategies may be necessary to avoid task dominance determined by the inherent scale and curvature of individual loss landscapes (like the one influenced by r in the NB case) and ensure balanced multi-objective optimization in joint training.

Another observation is that the joint models no longer generalize across negative binomial distributions as they did in the single model case. While this finding hints at sharper model selection, the fact that a more distinct distribution from poisson corresponding to a smaller r value is harder to learn, contradicts our intuition. As such, we leave this observation as an open question.



(a) Poisson Data

(b) Negative Binomial Data ($r = 1.2$)

(c) Negative Binomial Data ($r = 2.5$)

(d) Negative Binomial Data ($r = 5.0$)

Figure 4: **Joint training across different data distributions.** Each plot shows the validation loss of joint models on held-out Poisson and Negative Binomial datasets.

7 More Joint Training

Given our inability to learn the poisson and negative binomial (at r=1.2) distributions jointly, we aim to modify our training approach to see if is possible within our constraints.

7.1 Learning Rate and Curriculum Learning

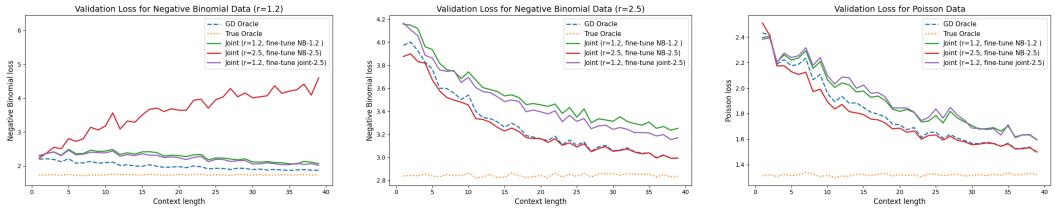
We ablate the learning rate and introduce a curriculum learning schedule to gradually increase task complexity. However, as shown in Appendix Figures 9 and 8, these modifications are to no avail: only the poisson task shows slight improvement, and neither configuration leads to convergence on both distributions. We leave this topic to the appendix due to its lack of interesting result.

7.2 Finetuning Pretrained Models

To investigate whether initialization from a pretrained model can improve joint training, we experiment with finetuning models trained on related distributions. For the challenging joint task involving the negative binomial ($r = 1.2$) and poisson data, we initialize from:

- A model trained solely on negative binomial data with $r = 2.5$,
- A model trained jointly on negative binomial ($r = 2.5$) and poisson data.

As a control, we also consider the case where we finetune the $r = 2.5$ joint model from a model trained solely on negative binomial data with $r = 2.5$.



(a) Negative Binomial with $r = 1.2$ (b) Negative Binomial with $r = 2.5$ (c) Poisson

Figure 5: **In-context Learning Joint Distributions with Finetuning.** We train a joint r=1.2 model initialized with a joint r=2.5 model and a negative binomial r=1.2 model. Additionally, we train a joint r=2.5 model initialized from a negative binomial r=2.5 model

In Figure 5, we observe that the joint task with $r = 1.2$ fails to improve under either initialization strategy, while the $r = 2.5$ joint task closely tracks oracle-level gradient descent when initialized from a pretrained model. Notably, the $r = 2.5$ joint reaches its final performance extremely early in training; within the first few hundred batches, as shown in Appendix Figure 10. These findings cast doubt on the presence of active model selection and instead suggest that the model quickly settles into a shared representation that fits both distributions, at least when they are sufficiently compatible. The precise mechanism behind this early convergence remains unclear as does the definitiveness of whether model selection is truly occurring.

8 Conclusion

In this work, we investigated whether transformer models can in-context learn various generalized linear models independently and jointly. We showed that the transformer architecture is not only capable of learning each of these GLMs individually through in-context learning, but is also able to jointly learn multiple GLM families in-context and distinguish between them in certain cases.

In the case of poisson and negative binomial regression, the models successfully adapted its predictions based on contextual dispersion patterns, implicitly selecting between models without access to ground truth task labels. This demonstrates that transformers can reason about underlying statistical structure and adjust their inference strategy accordingly, even when the distinction between function classes is non-trivial. However, we found limitations when the two learned function classes were too different, i.e negative binomial with $r = 1.2$ and poisson GLMs. We speculate on possible reasons why, but leave this phenomena as an open question.

These results extend the "transformer as a statistician" hypothesis and highlights the model's ability to internalize rich statistical behavior directly from in-context examples. The door is open to future work regarding in-context learning more complex probabilistic models, and further exploring what function classes are easiest to jointly learn and why.

9 Limitations and Future Work

All experiments were run on limited computational power (Nvidia A40), limiting the context window and model size tested. Since GLMs are asymptotically optimal, with only 40 data points maximum we may not observe the optimal GLM model in our comparisons to our individual and joint transformer models. The difficulties in jointly learning the poisson and negative binomial ($r = 1.2$) GLMs may be attributed to a lack of model expressivity or context size. However, our results support the general capability of transformers to learn GLM function classes with capability equal to that of traditional GLMs on limited datasets.

Furthermore, additional experiments may be necessary to explain certain results. In particular, we notice that while a negative binomial distribution with $r = 1.2$ is the most distinct from a poisson distribution and thus negative binomial with $r = 1.2$ contexts should be the "easiest" for a model to differentiate from poisson contexts, our joint model actually struggled most on $r = 1.2$.

A tempting conclusion to make from this is that our joint models are not actually learning to differentiate between GLMs, but instead mixes them, thus netting higher performance on negative

binomial-poisson combinations with the most similarity. However, our experiments on the individually trained models indicate that all of the negative binomial models are substantially different than the poisson model, indicating strong dissimilarities in the learned GLMs. This indicates that mixing may not explain the near-perfect performance in the case of jointly learning poisson data with negative binomial data using $r = 2.5, 5$.

An alternative conclusion is that the joint models are in fact learning model selection for $r = 2.5, 5$, but for some unknown reason fail on $r = 1.2$. This conclusion again faces the issue that intuitively and from prior work, substantially different function classes should be easier, not harder, to differentiate.

We believe future work is still required to fully understand the results presented above.

10 Appendix

10.1 Source Code + Model Weights

All source code is available at this repository, instructions are provided in the README. All train/eval plots are here as well.

`https://github.com/Logo96/in-context-learning-GLM.git`

Model weights are provided at this hugging face repository.

`https://huggingface.co/icl-182`

10.2 Major Peer Review Changes

- Added standardized evaluation plots for all models to enable direct comparison. Beforehand, our evaluation plots were not standardized.
- Added joint training experiment with $r = 2.5$ to study the effect of scaling on convergence and performance. We saw interesting results with $r = 1.2$ and $r = 5.0$ so we wanted to analyze this problem further.
- Conducted extensive analysis on the $r = 1.2$ poisson + NB joint task, including learning rate ablations, curriculum learning strategies, and finetuning from pretrained models.
- Removed linear, classification, and exponential experiments. We thought they were trivial and detracted from the main goal of our paper, that being model selection.

10.3 Exponential Training

During our experiments, we additionally trained a GPT-2 style transformer to perform in-context learning of an exponential GLM.

10.3.1 Exponential Regression

$$\begin{aligned} &\lambda = \exp(X\beta) \\ &Y \sim \text{Exponential}(\lambda) \\ &E[Y] = \frac{1}{\lambda} = \exp(-X\beta) \end{aligned}$$

The inverse-link function is $f(x) = \exp(-x)$, and the distribution of the response is exponential.

Note that the above formulation is technically non-canonical, but is more commonly used due to convenience and is also the form we train GPT-2 to in-context learn.

This model is appropriate for modeling positive and continuous data. It is especially useful for right-skewed data or the rate of occurrences.

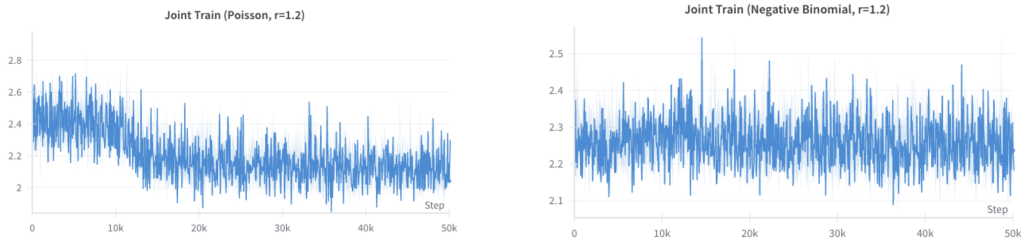
10.3.2 Exponential Training Process

Each output was sampled from an exponential distribution conditioned on the natural parameter, defined by $y_i \sim \text{Exponential}(\lambda_i)$ with $\lambda_i = \exp(x_i^\top w)$. The coefficient vector w was sampled from a gaussian prior $w \sim \mathcal{N}(0, I_{10})$, as in the linear regression setup. We again include a fixed scaling factor $\alpha = 0.32$

Training was conducted using the negative log-likelihood of the exponential distribution as the loss function, which corresponds to the mean negative log-likelihood under the exponential GLM.

Figure 6: **In-context learning of Exponential GLM.** Training losses.

10.4 Training Plots for Joint Training ($r = 1.2$)

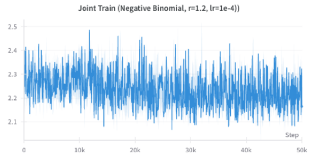


(a) Poisson Loss (Joint Training, $r = 1.2$)

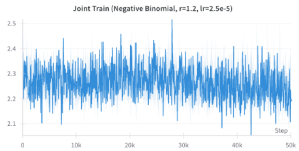
(b) Negative Binomial Loss (Joint Training, $r = 1.2$)

Figure 7: **Training loss with Negative Binomial $r = 1.2$ and Poisson data.** Here, we see neither curve converges to the known optimal (1.95 for NB-1.2 and 1.8 for poisson), and only the poison data is learned at all.

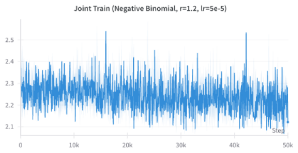
10.5 Learning Rate Ablation Plots



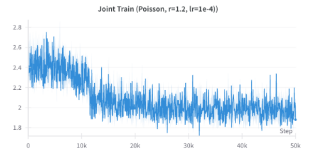
(a) Joint NB, $lr = 1e^{-4}$



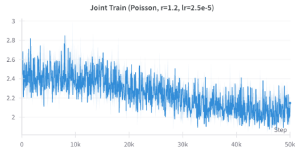
(b) Joint NB, $lr = 2.5e^{-5}$



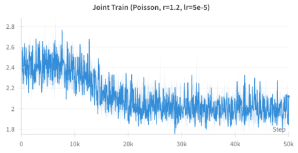
(c) Joint NB, $lr = 5e^{-5}$



(d) Joint Poisson, $lr = e^{-4}$



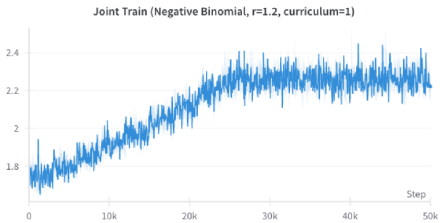
(e) Joint Poisson, $lr = 2.5e^{-5}$



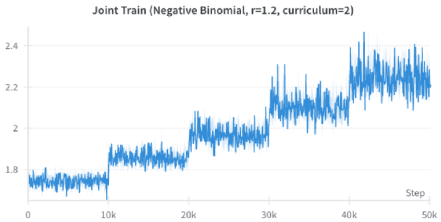
(f) Joint Poisson, $lr = 5e^{-5}$

Figure 8: Learning rate ablation for joint $r = 1.2$. We see that regardless of our learning rate, the model is unable to make progress on the negative binomial distribution. Furthermore, considering an ideal training loss of 1.8 or so from previous experiments, learning the poisson distribution is unoptimal here as well.

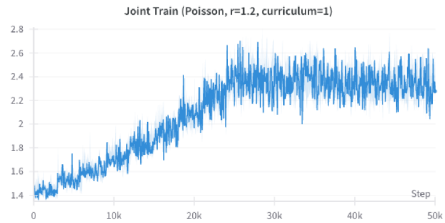
10.6 Curriculum Learning Ablation Plots



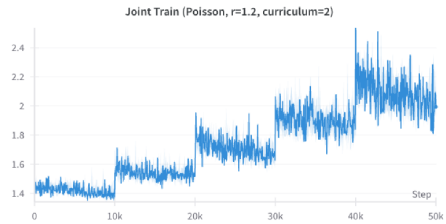
(a) NB curriculum rate = 1



(b) NB curriculum rate = 2



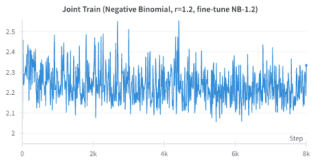
(c) Poisson curriculum rate = 1



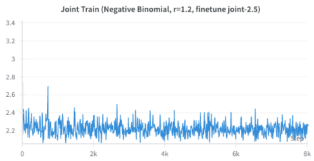
(d) Poisson curriculum rate = 2

Figure 9: Curriculum learning attempts for joint $r = 1.2$. We see that both the negative binomial and poisson distributions are not learned optimally where 1.95 and 1.8 are the optimal NLL loss from previous experiments.

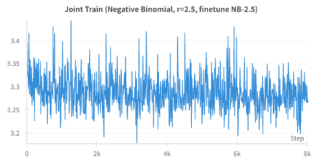
10.7 Finetuning Experiment Plots



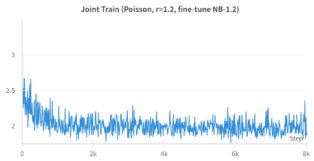
(a) NB Loss, Joint-1.2 (NB-1.2 base model)



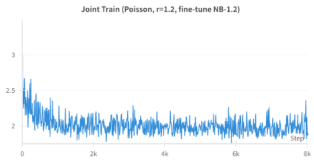
(b) NB Loss, Joint-1.2 (Joint-2.5 base model)



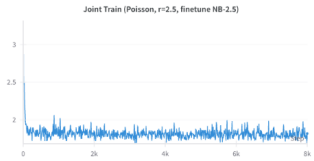
(c) NB Loss, Joint-2.5 (NB-2.5 base model)



(d) Poisson Loss, Joint-1.2 (NB-1.2 base model)



(e) Poisson Loss, Joint-1.2 (Joint-2.5 base model)



(f) Poisson Loss, Joint-2.5 (NB-2.5 base model)

Figure 10: Finetuning experiments train curves. We see that only the joint-2.5 model converges to ideal train loss, even when the base model is varied for the joint-1.2 model. Furthermore, we see rapid convergence for the joint-2.5 model (f). For reference, ideal train loss for poisson is 1.8, NB-1.2 is 1.95, and NB-2.5 is about 3.25 from previous experiments.

References

[1] Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection, 2023.

[2] Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes, 2023.

[3] Yingcong Li, M. Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning, 2023.

[4] Omar Naim and Nicholas Asher. Two in context learning tasks with complex functions, 2025.

[5] Anzo Teh, Mark Jabbour, and Yury Polyanskiy. Solving empirical bayes via transformers, 2025.