Dropout as a Regularizer of Interaction Effects

Benjamin Lengerich
MIT

Eric P. Xing
CMU, Petuum Inc., MBZUAI

Rich Caruana
Microsoft Research

Abstract

We examine Dropout through the perspective of *interactions*. This view provides a symmetry to explain Dropout: given N variables, there are $\binom{N}{k}$ possible sets of k variables to form an interaction (i.e. $\mathcal{O}(N^k)$); conversely, the probability an interaction of k variables survives Dropout at rate p is $(1-p)^k$ (decaying with k). These rates effectively cancel, and so Dropout regularizes against higher-order interactions. We prove this perspective analytically and empirically. This perspective of Dropout as a regularizer against interaction effects has several practical implications: (1) higher Dropout rates should be used when we need stronger regularization against spurious high-order interactions, (2) caution should be exercised when interpreting Dropout-based explanations and uncertainty measures, and (3) networks trained with Input Dropout are biased estimators. We also compare Dropout to other regularizers and find that it is difficult to obtain the same selective pressure against high-order interactions with these methods.

1 INTRODUCTION

We examine Dropout through the perspective of *interactions*: effects that require multiple variables. Given N variables, there are $\binom{N}{k}$ possible sets of k variables (N univariate effects, $\mathcal{O}(N^2)$ pairwise interactions, $\mathcal{O}(N^3)$ 3-way interactions); we can thus imagine that models with large representational capacity could be dominated by high-order interactions. In this paper, we show that Dropout contributes a regularization effect which helps neural networks (NNs) explore functions of lower-order interactions before considering functions of higher-order interactions. Dropout imposes this regularization by reducing the effective learning rate of higher-order interactions. As a result, Dropout encourages models to learn lower-order functions of additive components. This understanding of Dropout has implications for choosing Dropout rates: higher Dropout rates should be used when we need stronger regularization against interactions. This perspective also issues caution against using Dropout to measure term salience because Dropout regularizes against high-order interactions. Finally, this view of Dropout as a regularizer of interactions provides insight into the varying effectiveness of Dropout across architectures and datasets. We also compare Dropout to weight decay and early stopping and find that it is difficult to obtain the same regularization with these alternatives.

Why Interaction Effects? Dropout was originally motivated to prevent "complex co-adaptations in which a feature detector is only helpful in the context of several other specific feature detectors" (Hinton et al., 2012; Srivastava et al., 2014). Most "complex co-adaptations" are interaction effects, so it is natural to quantify the effect of Dropout through the lens of interaction effects. This perspective is valuable because (1) modern NNs, containing intractable numbers of parameters, are more suitable to analysis via nonparametric functional analysis than parametric analysis, (2) interaction effects can be calculated identifiably, and (3) this perspective has practical implications on choosing Dropout rates. To preview the results, when NNs are trained on data without important interactions, the optimal Dropout rate is high, but when NNs are trained on data with important interaction effects, the optimal Dropout rate is lower.

2 PRELIMINARIES AND RELATED WORK

2.1 Functional ANOVA and Pure Interaction Effects

In this paper, we use the concept of *pure interaction effects*: a pure interaction effect is variance explained by

a group of variables u that *cannot* be explained by any group of variables u' where $|u'| < |u|$ (e.g. any subset of u) (Lengerich et al., 2020). Multiplicative terms like $X_1 X_2$ are often used to encode "interaction effects". They are, however, only *pure* interaction effects if X_1 and X_2 are uncorrelated and have mean zero; otherwise, some portion of the variance in the outcome $X_1 X_2$ could be explained by main effects of each individual variable. Correlation between two variables does not imply an interaction effect on the outcome, and an interaction effect of two variables on the outcome does not imply correlation between the variables.

This definition of pure interaction effects is equivalent to the functional ANOVA (fANOVA) decomposition: Given a density $w(X)$ and a family of allowable functions $\mathcal{F}^u$ for variable set u, where $\mathcal{F}^u \subset \mathcal{L}^2(\mathbb{R}^u)$ the set of square-integrable functions, the weighted fANOVA (Hooker, 2004, 2007; Cuevas et al., 2004) decomposition of $F(X)$ is:

$$\{f_u(X_u)|u \subseteq [d]\} = \tag{1a}$$

$$\underset{\{g_u \in \mathcal{F}^u\}_{u \in [d]}}{\arg\min} \int \Big(\sum_{u \subseteq [d]} g_u(X_u) - F(X) \Big)^2 w(X) dX, \tag{1b}$$

such that

$$\forall\, v \subseteq\, u, \quad \int f_u(X_u) g_v(X_v) w(X) dX = 0 \quad \forall\, g_v, \tag{1c}$$

where $[d]$ indicates the power set of d features. This decomposition defines each member f_u to be orthogonal to the members which operate on any subset of u.[1] An interaction effect f_u is of *order k* if $|u| = k$. Given N variables, there are $\binom{N}{k}$ possible sets of size k, so we say that there are $\binom{N}{k}$ interaction effects of order k.

2.2 Related Work

Hinton et al. proposed Dropout to prevent spurious co-adaptation (i.e., spurious interactions), and it has proved an extremely effective regularizer of deep models. However, questions remain. For example: Is the expectation of the output of a NN trained with Dropout the same as for a NN trained without Dropout? Does Dropout change the trajectory of learning during optimization even in the asymptotic limit of infinite training data? Should Dropout be used at run-time when

---

[1] The fANOVA decomposition describes a unique decomposition for a given data distribution; thus, pure interaction effects are defined in conjunction with a data distribution. An example of this interplay between is shown in Figure A.1. As Lengerich et al. (2020) describe, the correct distribution to use is the data-generating distribution $p(x)$. Estimating $p(x)$ is one of the central challenges of machine learning; for this paper, we mainly use simulation data for which we know $p(x)$.

querying a NN to see what it has learned? These questions are important because Dropout has been used as a method for Bayesian uncertainty (Gal and Ghahramani, 2016; Gal et al., 2017; Chang et al., 2017b,a). The use of Dropout for uncertainty quantification has been questioned due to its failure to separate aleotoric and epistemic sources of uncertainty (Osband, 2016). In this paper we ask a separate yet related question: Does Dropout treat functions equivalently?

Significant work has focused on the effect of Dropout as a weight regularizer (Baldi and Sadowski, 2013; Warde-Farley et al., 2013; Cavazza et al., 2018; Mianjy et al., 2018; Zunino et al., 2018), including its properties of structured shrinkage (Nalisnick et al., 2018) or adaptive regularization (Wager et al., 2013). However, weight regularization is of limited utility for modern-scale NNs. Instead of focusing on the influence of Dropout on parameters, we take a nonparametric view of NNs as function approximators. Thus, our work is similar in spirit to Wan et al. (2013), which showed a linear relationship between keep probability and Rademacher complexity, and Duvenaud et al. (2014), which showed that Dropout can be viewed as a mixture of models which each depend on only a subset of the input variables. Our work crystallizes these observations into a description of Dropout as a regularizer of interaction effects, resulting in models that generalize better by down-weighting high-order interaction effects.

3 ANALYSIS: DROPOUT REGULARIZES INTERACTION EFFECTS

Dropout operates by probabilistically setting values to zero (i.e. multiplying by a Bernoulli mask). For clarity, we call this operation "Input Dropout" if the perturbed values are input variables, and "Activation Dropout" if the perturbed values are activations of hidden nodes.

Input Dropout Shrinks Interaction Effects First, Input Dropout replaces the training dataset with samples drawn from a perturbed distribution:

Theorem 1. *Let* $\mathbb{E}[Y|X] = F(X) = \sum_{u \in [d]} f_u(X_u)$ *and* $\tilde{Y} = F(X \odot M)$, *where* $M_p \sim Bernoulli(p)$ *is the Input Dropout mask and* $\odot$ *is element-wise multiplication. Then*

$$\mathbb{E}[\tilde{Y}|X] = \sum_{u \in [d]} (1-p)^{|u|} f_u(X) + a_u(X_u). \tag{2}$$

with $a_u(X_u) = \sum_{v \subseteq u} p^{|u|-|v|} (1-p)^{|v|} f_u(X_{u \setminus v}, X_v = 0)$.

This theorem shows that Input Dropout shrinks the conditional expectation of $\tilde{Y}|X$ by preferentially targeting high-order interactions: the scaling factor $(1-p)^{|u|}$

shrinks exponentially with $|u|$. For multiplicative interaction effects, we can simplify $a_u(X_u)$:

Corollary 1.1. *Let $f_u(X_u)$ be a multiplicative interaction effect such that $f_u(X_u) = \prod_{u' \in u} g_{u'}(X_{u'})$ with $g_{u'}(0) = 0$ for each subset $u' \in u$. Then*

$$\mathbb{E}[\tilde{Y}|X] = \sum_{u \in [d]} (1-p)^{|u|} f_u(X). \tag{3}$$

Proof. Let $f_u(X_u)$ be a multiplicative effect. Then

$$f_u(X_{u \setminus v}, X_v = 0) = 0 \quad \forall \, v \, \in \, u \tag{4}$$

and hence $a_u(X_u) = 0 \, \forall \, u$. □

This theorem implies that for multiplicative interaction effects, the shrinkage factor is exact. We can visualize this order-specific shrinkage of Input Dropout by empirically calculating $\mathbb{E}[\tilde{Y}|X]$ for $\tilde{Y} = f_u(X_u \odot M)$. In Figure 1, we examine four multiplicative interaction effects: $f_u(X) = X_1$ for $k = 1$, $f_u(X) = X_1 X_2$ for $k = 2$, $f_u(X) = X_1 X_2 X_3$ for $k = 3$, and $f_u(X) = X_1 X_2 X_3 X_4$ for $k = 4$. For each of these interaction orders, we measure $\mathbb{E}[\tilde{Y}|X]$ for various input Dropout rates. We see that the shrinkage effect of Input Dropout is strongest for $k = 4$ (e.g. the conditional mean observed under Input Dropout of $p = 0.5$ is 12.5% of the conditional mean observed without Input Dropout), and weakest for $k = 1$ (e.g. the conditional mean observed under Input Dropout of $p = 0.5$ is 50% of the conditional mean observed without input Dropout).

Thus, the conditional expectation of the outcome changes with Input Dropout, and the change is dependent on the order of the interaction effect. In practice, this means that Input Dropout changes the distribution of model predictions, so even NNs trained for more epochs or with large sample size will converge to different optima based on the Input Dropout level. This is unlike ℓ_1 or ℓ_2 weight regularization which can be overcome by increasing the size of the training set and are affected by the downstream net architecture. Finally, Input Dropout affects higher-order interactions more than lower-order interactions, biasing the prediction of any model (regardless of model training procedure).

Dropout Shrinks Gradients Next, we examine how Dropout affects training:

Corollary 1.2. *Let $G(X, Y, \theta) = \nabla_\theta(f_\theta(X), Y)$ be the gradient update from data X, Y, with $G(X, Y, \theta) = \sum_u g_u(X_u, Y, \theta)$ the fANOVA decomposition with respect to X. Then for mask $M_p \sim Bern(p)$,*

$$\mathbb{E}[g_u(X_u \odot M_p, Y, \theta)] = \tag{5}$$
$$(1-p)^{|u|} g_u(X_u, Y, \theta) + a_u(X_u, Y, \theta)$$

where $\mathbb{E}_{X,Y,\theta}[a_u(X_u, Y, \theta)] = 0$.

(a) $k = 1$ (b) $k = 2$
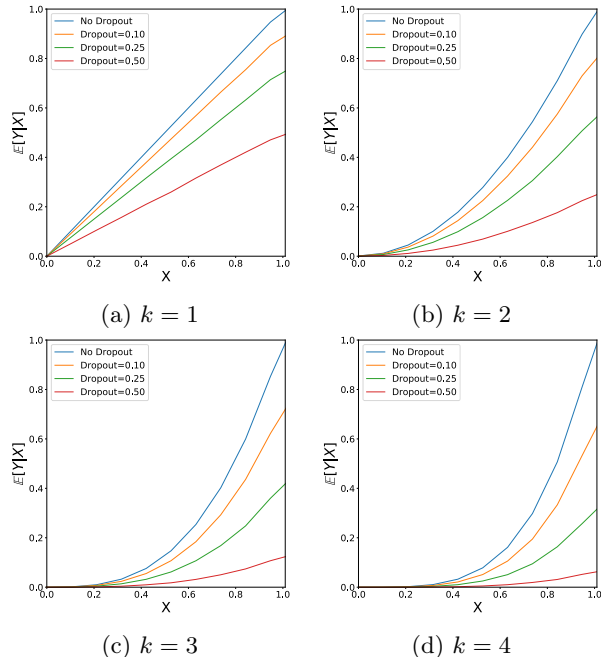
(c) $k = 3$ (d) $k = 4$

Figure 1: Visualizing the impact of Input Dropout on observed conditional means. we visualize the outcome which is a fixed multiplicative interaction effect of order k. Each colored line provides the conditional expectation of the outcome observed under a particular frequency of Input Dropout. As k increases, the shrinkage effect of Input Dropout increases.

Proof. Let $F(X) = G(X, Y, \theta)$ for fixed Y, θ. Then $\mathbb{E}[F(X \odot M_p)] = \sum_{u \in [d]} (1-p)^{|u|} f_u(X) + a_u(X_u)$ by Theorem 1. □

That is, Input Dropout shrinks gradient updates according to the order of the interaction effect which produced the gradient updates. We can see this effect by examining the first gradient (at initialization) of NNs calculated under various levels of input Dropout. In Figure. 2, we show the distribution of the ℓ_1-norm norm of these gradients normalized by the ℓ_1-norm of the gradients for a NN without input Dropout. We see that for NNs fitting to an interaction effect of order $k = 1$ (Fig. 2a), the gradients are shrunk only slightly under large Dropout, while for NNs fitting to an interaction effect of order $k = 4$ (Fig. 2d), the gradients are shrunk severely under large Dropout.

To describe Activation Dropout, we modify G to act on activations of a particular layer:

Corollary 1.3. *Let $G_i(A_i, Y, \theta) = \nabla_\theta(f_\theta(A_i), Y)$ be the gradient update for parameters θ from activation A_i at layer i with target outcome Y. Let $G_i(A_i, Y, \theta) = \sum_u g_{iu}(A_{iu}, Y, \theta)$ be the fANOVA decomposition of $G_i(A_i, Y, \theta)$ with respect to A. Then for mask $M_p \sim$*
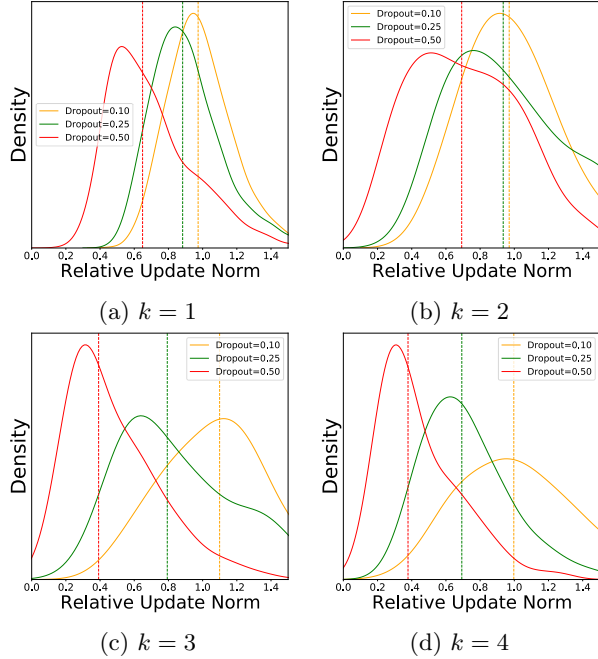
(a) $k = 1$

(b) $k = 2$

(c) $k = 3$

(d) $k = 4$

Figure 2: Distribution of gradient magnitudes during training on various orders of interaction effect ($k = 1, 2, 3,$ or 4) with various levels of Dropout. We normalize the gradient norms by comparing to the gradient induced by training for the same training points and initialization with a Dropout rate of 0.0. Dashed vertical lines indicate the median of each distribution (best viewed in color). The gradient norms are down-weighted for higher-interaction effects and for higher Dropout rates. For more details about this experiment, please refer to Corollary 1.2.

$Bern(p),$

$$\mathbb{E}[g_{iuvw}(A_{iu} \odot M_p, Y, \theta)] = \qquad (6)$$
$$(1-p)^{|u|} g_{iu}(A_{iu}, Y, \theta) + a_u(A_{iu}, Y, \theta)$$

where $\mathbb{E}_{A_i, Y, \theta}[a_u(A_{iu}, Y, \theta)] = 0.$

Proof. Let $F(A_i) = G_i(A_i, Y, \theta)$ for fixed Y, θ. Then $\mathbb{E}[F(A_i \odot M_p)] = \sum_{u \in [d]} (1-p)^{|u|} f_u(X) + a_u(X_u)$ by Theorem 1. □

Activation Dropout thus shrinks the gradient update according to the number of hidden nodes which interact to form the gradient update. Thus, both Input Dropout and Activation Dropout correspond to *effective learning rates* $r_p(k) = (1-p)^k$ (the distinction being whether k counts the *input features* in the interaction or counts the *hidden activations* in the interaction)[2].

---

[2]This explains why Dropout tends to produce hidden units which are "specialized" (Gigante et al., 2019).

Symmetry Between Dropout Strength and Number of Interaction Effects The effective learning rate $r_p(k) = (1-p)^k$ of k-order interactions decays exponentially with k. This is a symmetry with $\binom{N}{k}$. As shown in Fig 3, the exponential growth of the hypothesis space $|\mathcal{H}_k| = \binom{N}{k}$ is balanced by the exponential decay of the effective learning rate.
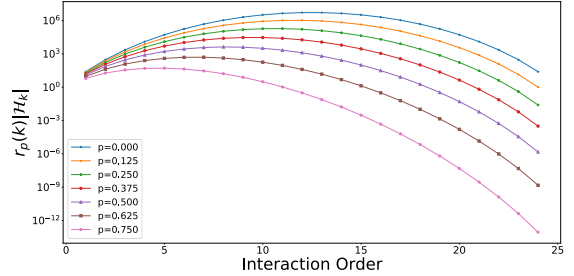
(a) Full space.
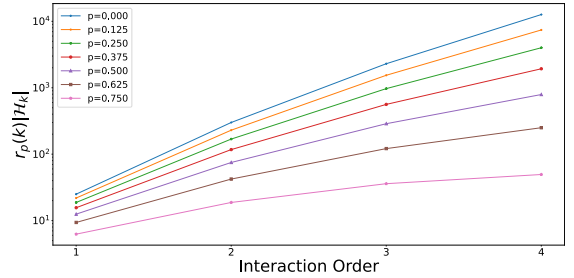
(b) First four orders.

Figure 3: The growing hypothesis space of interaction effects is balanced against the effective learning rate imposed by Dropout. In this figure, we plot the product of the effective learning rate ($r_p(k)$) and the number of potential interaction effects of order k ($|\mathcal{H}_k|$). In a, we plot these values on a log scale for the entire range of interaction orders for an input of $N = 25$ features. In b, we plot up to order 4.

4 EXPERIMENTS

As shown above, Dropout exerts regularization against interaction effects. Here, we decompose NNs to measure the strength of learned interaction effects.

4.1 Measuring Interaction Effects in NNs

As with any function, the $\hat{F}(X)$ learned by a NN can be decomposed as: $\hat{F}(X) = \sum_{u \in [d]} \hat{f}_u(X_u)$ by fANOVA (Eq. 1c). To calculate this decomposition, we apply model distillation Buciluǎ et al. (2006); Hinton et al. (2015) using the XGBoost software package Chen and Guestrin (2016) to train boosted decision trees with maximum depth k to distill the interaction effects of order k. By successively increasing k and training on the

residuals of the shallower trees, the estimated effects are orthogonal and hence satisfy fANOVA requirements.

Accuracy of Decomposition How accurate is this distillation procedure for calculating the fANOVA decomposition of a NN? To empirically validate this procedure, we test distillation using simulation data. Our goal in this experiment is to measure the error of distillation on NNs representing known functions.[3] For each run, we generate data according to $X \sim \text{Unif}(-1,1)^5$, and train a NN to fit a pure k-order interaction (a multiplication of k uncorrelated features of X). A perfect distillation procedure would assign 100% of the variance to interactions of order k.

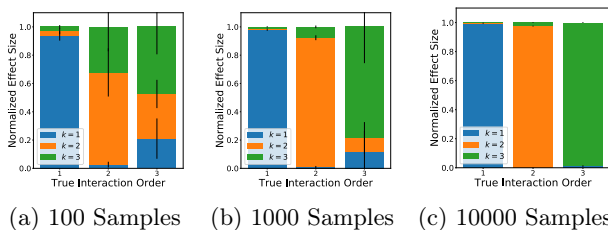(a) 100 Samples   (b) 1000 Samples   (c) 10000 Samples

Figure 4: Distilled effect sizes (mean $\pm$ var over 10 runs) of the interactions in NNs representing interactions of order 1, 2, or 3. The size of effect f_X is defined as $\text{Var}_X(f_X(X))$. Each pane shows results for a number of samples used for distillation.

Results are shown in Fig. 4. In each pane, there are 3 bars which each represent an order. The height of the bars (and the corresponding colors) represent the normalized effect size estimated by the distillation procedure. In Fig. 4a, only 100 samples are used for distilling the trained NN into its composite interaction effects; as a result, the low-order models underfit the NN and exaggerate the effects of high-order interactions. When the number of samples is increased to 1000 (Fig. 4b) or to 10000 (Fig. 4c), the distillation procedure is increasingly accurate at recovering the true interaction breakdown in the NN. Importantly, none of the distillations over-estimated the influence of low-order interaction effects.

---

[3]We do not claim that this distillation is always suitable as general-purpose explanations of NNs; in this context, however, we care about only a single aspect of the compressed models: approximation error. For example, from the NN $\hat{F}(X)$, we estimate an additive model $\hat{f}_1(X) \in \mathcal{S}(\hat{F}, \mathcal{F}_1) = \arg\min_{f \in \mathcal{F}_1} \mathbb{E}_X[(f(X) - \hat{F}(X))^2]$ where $\mathcal{F}_1$ is the class of additive models. The set of possible explanations $\mathcal{S}(\hat{F}, \mathcal{F}_1)$ may have more than one member; however, all of these explanations must have the same compression loss. As the only metric we are reporting about these models is the compression loss, all members of $\mathcal{S}(\hat{F}, \mathcal{F}_1)$ are equivalent.

4.2 Dropout Regularizes Spurious Interactions

In this experiment, we use a simulation setting in which there is no signal (so any estimated effects are spurious). This gives us a testbench to easily see the regularization strength of different levels of Dropout. Specially, we generate 1500 samples of 25 input features where $X_i \sim \text{Unif}(-1,1)$ and $Y \sim N(0,1)$. We optimize NNs with 3 hidden layers and ReLU nonlinearities and measure effect sizes as described in Sec. 4.1. In Fig. 5, we see the results for NNs with 32 units in each hidden layer. For this small network, both Activation and Input Dropout have strong regularizing effects on a NN. Not only do they reduce the overall estimated effect size, both Activation and Input Dropout preferentially target higher-order interactions (e.g., the proportion of variance explained by low-order interactions monotonically increases as the Dropout Rate is increased for Figs. 5d,5e, and 5f. In Fig. C.1, we see results from the same experiment on NNs with 128 units in each hidden layer; as our analysis predicts, Input Dropout is just as strong for this network (Fig. C.1e).

4.3 Optimal Dropout Rate Depends On True Interactions

A natural application of this perspective is that Dropout should be used at higher rates where we need to regularize against interaction effects. To test this guideline, we perform two experiments.

Modified 20-NewsGroups Data We use the 20-NewsGroups dataset [4], which is a classification task on documents from 20 news organizations. We modify this dataset by adding k new features (each feature is IID Unif$(0,1)$) and a 21st class which is the correct label if all of the k new features take on a value greater than 0.5. This modified dataset then has a strong k-way interaction effect, and as k grows, we would expect the optimal Dropout rate to be lower. As predicted by our understanding of Dropout, indeed the optimal Dropout rate is lower for larger k; with optimal rates of 0.375 for $k = 1$, 0.25 for $k = 2$, and 0.125 for $k = 3$ (full results are shown in Table C.1).

BikeShare The New York City BikeShare dataset[5] (preprocessing from[6]) is a large dataset designed to help predict the demand of Citi Bikes in New York City. Because bicyclists base travel plans on hourly, daily, and weekly cycles, there are real interaction effects in this dataset (Tan et al., 2018). As predicted by the

---

[4]http://qwone.com/~jason/20Newsgroups/
[5]https://www.citibikenyc.com/system-data
[6]https://www.kaggle.com/akkithetechie/new-york-city-bike-share-dataset

(a) Total: Activation

(b) Total: Input

(c) Total: Input + Act.

(d) Normalized: Activation
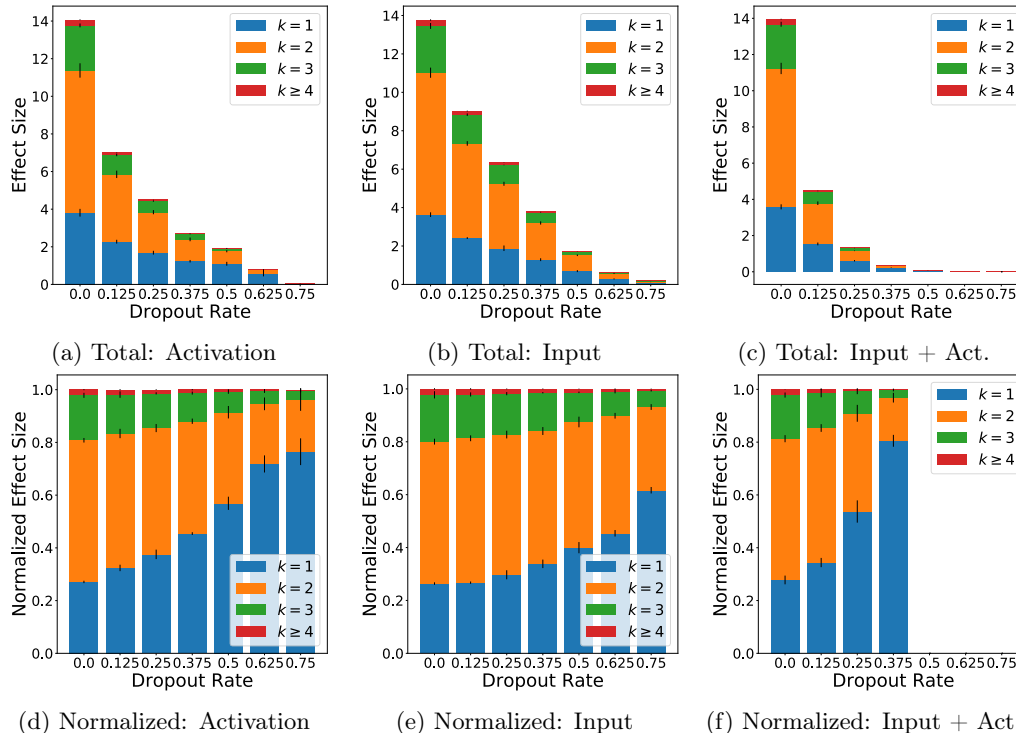
(e) Normalized: Input

(f) Normalized: Input + Act.

Figure 5: NNs trained on pure noise (details in Sec. 4.2). Displayed values are the mean ± std. over 10 runs for each interaction order. Activation and Input Dropout both reduce the effect sizes of the learned high-order interactions. The top row (a–c) shows absolute effect sizes (which decrease as Dropout increases), while the middle row (d–f) shows the relative effect sizes, making it easier to see how the Dropout rate affects each order.

interaction view of Dropout, the optimal Dropout rate Dropout is 0 (full results in Fig. 6).

4.4 Do Other Regularizers Penalize Interaction Effects?

Here, we examine early stopping and weight decay as potential regularizers of interaction effects. We find that neither of these regularization techniques specifically target interaction effects. However, because Dropout changes the effective learning rate of interaction effects, it can act in concert with early stopping to magnify the regularization against interaction effects.

Early Stopping The effective capacity of NNs increases during training (Weigend, 1994), and recent work supports the view that randomly-initialized NNs start as simple functions that are made more complex through training (De Palma et al., 2018; Nakkiran et al., 2019; Jacot et al., 2018). Thus, it makes sense that early stopping can help select models that generalize well (Prechelt, 1998; Caruana et al., 2001). To see how early stopping interplays with the Dropout-induced effective learning rates, we study the effects learned over the course of optimization.

We generate 1500 samples of 25 input features where $X_i \sim \text{Unif}(-1, 1)$ and the target is generated according to one of three settings: (1) only main effects: $Y \sim \text{N}(\sin(X_0) + \cos(X_1), \sigma^2)$, (2) only pair effects: $Y \sim \text{N}(\sin(X_0)\cos(X_1), \sigma^2)$, and (3) only three-way effects: $Y \sim \text{N}(\sin(X_0)\cos(X_1)X_2, \sigma^2)$. We optimize fully-connected NNs on these data and measure effect sizes as described in Sec. 4.1. Results are shown in Fig. 7. The key findings are: 1) the rightmost column shows that NNs with low rates of Dropout tend to massively overfit due to a reliance on high-order interactions; 2) the different levels of Dropout have different steady-state optima; 3) because Dropout slows the learning of high-order effects, early stopping is doubly effective in combination with Dropout. NNs tend to learn simple functions earlier (regardless of Dropout usage), and Dropout slows the learning of high-order interactions; these factors combine to reduce the complexity of the learned function under early stopping.

Weight Decay Another popular regularization mechanism is weight decay: placing an ℓ_2 penalty on the weights of the network. We study weight decay on the same data generator as we studied Dropout in Sec. 4.2. As the results in Fig. 8 show, strong weight decay (large values of λ) has a modest effect of regu-
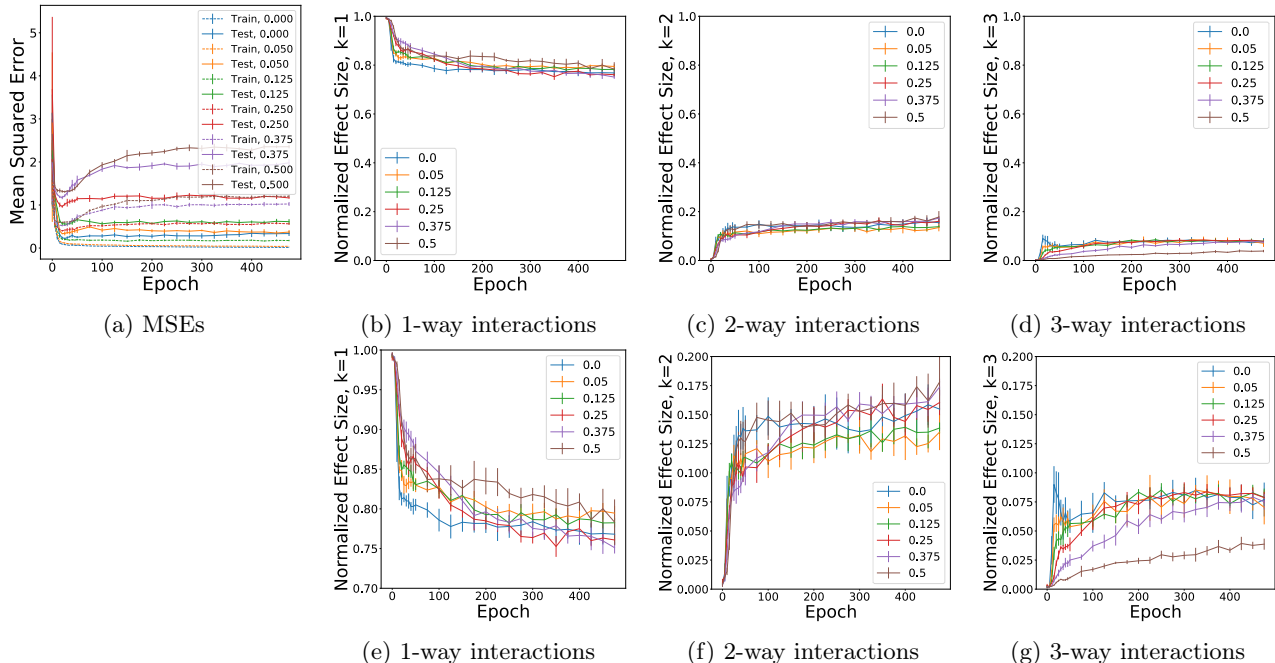
Benjamin Lengerich, Eric P. Xing, Rich Caruana

Figure 6: Learned interaction effects and model errors over epochs training on the BikeShare Dataset. (a) Mean squared errors by epoch of models trained with varying rates of Dropout. In this dataset, there are true interaction effects of orders 2 and 3, so the models with high Dropout rates generalize *worse* than the models with low Dropout rates. This behavior is expected under our perspective of Dropout as an interaction regularizer, but would be unexpected under the perspective of Dropout as an unbiased regularizer. (b-d) Normalized effect sizes by epoch learned in NNs with varying levels of Dropout. (e-g) Normalized effect sizes, with values repeated from panes b-d, but vertical axes scaled to show relative differences. We see that even small differences in the absolute magnitude of effect size for each interaction order can produce large differences in model generalizability.

larizing against interaction effects. However, achieving the same practical benefit from weight decay as from Dropout is untenable due to the training instability that strong weight decay introduces: when weight decay was larger than 0.2, the NNs learned constant functions.

5 DISCUSSION AND IMPLICATIONS

In this paper, we examined a concrete mechanistic explanation of how Dropout works: by regularizing higher-order interactions. This explanation of Dropout has several implications for its use and crystallizes some of the conventional wisdom regarding how and when to use Dropout.

5.1 Dropout For Explanations

While Dropout has been used for measures of model confidence (Gal and Ghahramani, 2016; Gal et al., 2017) and to aid model interpretability (Chang et al., 2017b,a), it does not treat all effects equally. This bias is present both during estimation and inference. We examine the distributions of predictions and uncer-

tainties produced by NNs under various Dropout rates (Fig. 9). In this experiment, we train NNs to predict a random variable which is the product of k uncorrelated Bernoulli variables. We generate sufficient samples that all NNs learn to confidently predict the outcome for all orders (Fig. 9a). However, when Dropout is used (Fig. 9b), the models equivocate, with much greater uncertainty for the higher-order interaction effects – the model representing an interaction of order 1 is barely affected, while the model representing an interaction of order 4 is ambivalent. Thus, the bias of Dropout is reflected when measuring model confidence.

5.2 Setting Dropout Rate

The Dropout rate should be set according to the desired magnitude of the anti-interaction regularization effect. If the dataset is large or sufficient augmentation can be performed, lower rates of Dropout can be used or Dropout can be omitted entirely (e.g. the New York City BikeShare dataset discussed in Section 4.3). In addition, it is often suggested to use larger Dropout rates in deeper layers than in initial layers (Ba and Frey, 2013). This wisdom can be explained from the interac-
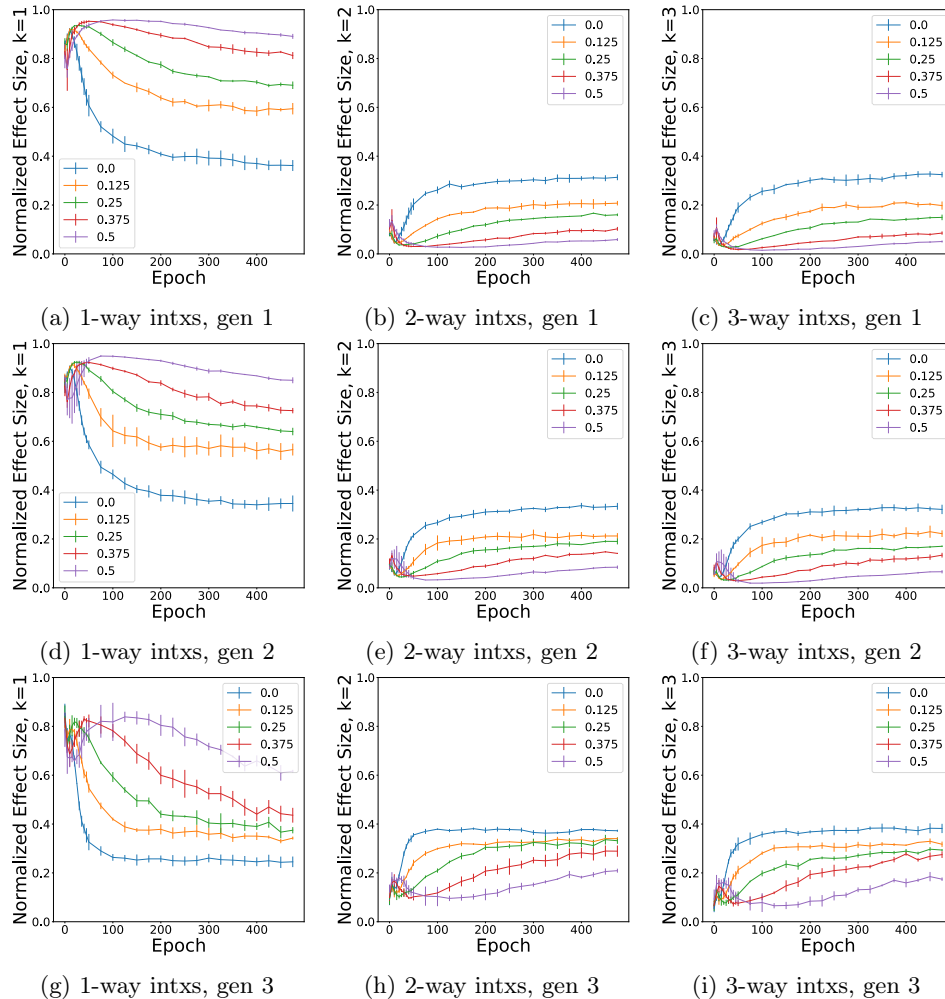
Figure 7: Learned interaction effects of order 1, 2 and 3 (cols 1, 2, and 3 respectively) by epoch. Each row corresponds to a different generator as described in Sec. 4.4: the generator in the top row has only 1-way interactions, the generator in the middle row has only 2-way interactions, and the bottom row has only true 3-way interactions. Key findings are described in Sec. 4.4.

tion perspective: this regularization scheme encourages NNs to do representation learning, which may require learning interactions between input features such as pixels or words, in the initial layers, while encouraging deeper layers to focus more on summing evidence from multiple sources.

In CNNs, Dropout is typically used at lower rates than in fully-connected networks (Park and Kwak, 2016). The convolutional architecture creates constraints that prevent arbitrary high-order interactions by restricting N in $\binom{N}{k}$ to be a selected set of local input features or hidden unit activations. Operators like max pooling further restrict the model's ability to learn complex interactions. In other words, convolutional nets create a strong architectural bias for or against different kinds of interaction effects and thus depend less on a mechanism like Dropout to blindly regularize interactions.

5.3 Explicitly Modeling Interaction Effects

A major challenge of estimating interaction effects is the hypothesis space which grows exponentially with the order of the interaction effect. If we were able to reduce the hypothesis space by specifying a small set of potential interaction effects a priori, our models could efficiently learn the correct parameters for these few interactions from data. Several recent works have proposed to do this by explicitly specifying the interaction effects the NNs may consider. Of particular note is (Jayakumar et al., 2020), which proposed to use multiplicative interactions to combine data modalities, and found that many common architectures can be seen in the lens of multiplicative interactions.

Another approach to explicitly model interaction effects is the Deep and Cross Network (Wang et al., 2017),

Benjamin Lengerich, Eric P. Xing, Rich Caruana

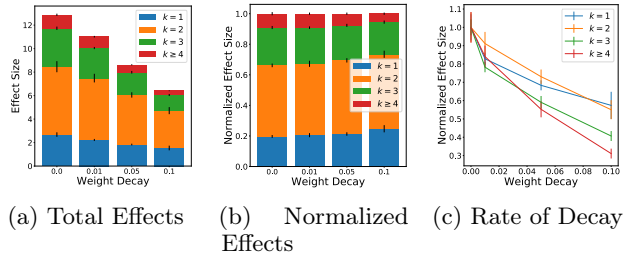(a) Total Effects   (b) Normalized Effects   (c) Rate of Decay

Figure 8: Weight decay can weakly regularize against interactions; however, the regularization comparable to Dropout occurs at extremely strong weight decay for which training is unstable.

which uses a two-part architecture consisting of a fully-connected network and a "cross" network in which each layer has its activation crossed with the vector of input variables, increasing the interaction order at every layer. Interestingly, the experiments of Wang et al. (2017) (especially Fig. 3 within) show that the best-performing architecture has only a single cross layer – exactly what we would expect from the amount of spurious interaction effects NNs are capable of learning.
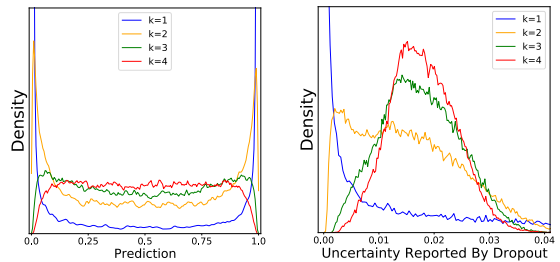
5.4 Impure Interaction Effects

We have used pure interaction effects as a lens through which to analyze the impact of Dropout; but what effect does Dropout have on *impure* interaction effects? The analysis of pure interaction effects naturally extends to impure interaction effects because all functions, even non-pure interactions, can be written as a combination of pure interaction effects. We look forward to future work examining the impact of Dropout on functions which are often written as non-pure interaction effects.

5.5 Limitations and Broader Impacts

This new perspective on Drooput as a regularizer of interactions effects helps to crystallize Dropout use cases and guidelines, but is in no way a full picture of NN behavior. There are many reasons why over-parameterized models such as deep NNs generalize to unseen data; here we have explored only one of the contributing factors of a regularizer against spurious interaction effects. Nevertheless, we believe that theoretical insights and concise description of NN behaviors, such as this perspective on Dropout, can provide broader impacts driven by more precise descriptions of machine learning behavior. Without precise theoretical understanding, significant resources must be invested to hyperparameter and architecture tuning; better theoretical understanding can direct system design and reduce burdensome resource requirements.
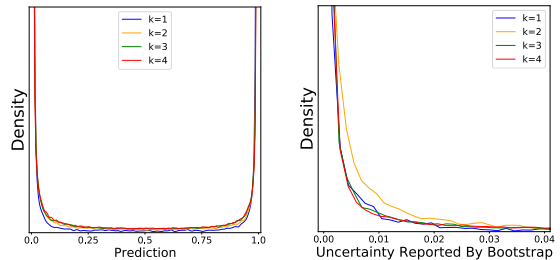
(a) Histogram of predictions (left) and uncertainties (right) without Dropout

(b) Histogram of predictions (left) and uncertainties (right) with Dropout at $p = 0.25$

Figure 9: Visualizing the predictions and uncertainties measured by bootstrap (a) and Dropout (b). Dropout targets higher-order interaction effects, leading to a bias in the reported uncertainties. Uncertainty is defined as the variance of the prediction over all Dropout masks.

6 CONCLUSIONS

In this paper, we have examined a concrete explanation of Dropout as a regularization against interaction effects. We have shown that the effective learning rate of interaction effects decreases exponentially with the order of the interaction effect, a crucial balance against the exponentially-growing number of potential interactions of k variables. Input Dropout targets interactions of k input variables, while Activation Dropout targets interactions of k hidden units in a layer. Although Dropout can work in concert with weight decay and early stopping, these do not naturally achieve Dropout's regularization against high-order interactions. By reducing the tendency of NNs to learn spurious high-order interaction effects, Dropout helps to train models which generalize more accurately to test sets.

References

J. Ba and B. Frey. Adaptive dropout for training deep neural networks. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3084–3092. Curran Associates, Inc., 2013.

P. Baldi and P. J. Sadowski. Understanding dropout. In *Advances in neural information processing systems*, pages 2814–2822, 2013.

C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

R. Caruana, S. Lawrence, and C. L. Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408, 2001.

J. Cavazza, P. Morerio, B. Haeffele, C. Lane, V. Murino, and R. Vidal. Dropout as a low-rank regularizer for matrix factorization. In *International Conference on Artificial Intelligence and Statistics*, pages 435–444, 2018.

C.-H. Chang, E. Creager, A. Goldenberg, and D. Duvenaud. Interpreting neural network classifications with variational dropout saliency maps. In *NeurIPS Machine Learning in Computational Biology Workshop*, 2017a.

C.-H. Chang, L. Rampasek, and A. Goldenberg. Dropout feature ranking for deep learning models. *arXiv preprint arXiv:1712.08645*, 2017b.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. *ArXiv*, abs/1603.02754, 2016.

X. Cheng, B. Khomtchouk, N. Matloff, and P. Mohanty. Polynomial regression as an alternative to neural nets. *CoRR*, abs/1806.06850, 2018.

B. A. Coull, D. Ruppert, and M. Wand. Simple incorporation of interactions into additive models. *Biometrics*, 57 (2):539–545, 2001.

A. Cuevas, M. Febrero, and R. Fraiman. An anova test for functional data. *Computational statistics & data analysis*, 47(1):111–122, 2004.

G. De Palma, B. T. Kiani, and S. Lloyd. Deep neural networks are biased towards simple functions. *arXiv preprint arXiv:1812.10156*, 2018.

I. A. Delbridge, D. S. Bindel, and A. G. Wilson. Randomly projected additive gaussian processes for regression. *arXiv preprint arXiv:1912.12834*, 2019.

D. Duvenaud, O. Rippel, R. Adams, and Z. Ghahramani. Avoiding pathologies in very deep networks. In *Artificial Intelligence and Statistics*, pages 202–210. PMLR, 2014.

P. H. Eilers and B. D. Marx. Flexible smoothing with b-splines and penalties. *Statistical science*, pages 89–102, 1996.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

Y. Gal, J. Hron, and A. Kendall. Concrete dropout. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3581–3590. Curran Associates, Inc., 2017.

A. Gelman. Statistical modeling, causal inference, and social science, Mar 2018.

S. Gigante, A. S. Charles, S. Krishnaswamy, and G. Mishne. Visualizing the phate of neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 1840–1851. Curran Associates, Inc., 2019.

T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall/CRC, 1990.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

G. Hooker. *Diagnostics and Extrapolation in Machine Learning*. Stanford University, 2004.

G. Hooker. Generalized functional anova diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics*, 16(3): 709–732, 2007.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8571–8580. Curran Associates, Inc., 2018.

S. M. Jayakumar, J. Menick, W. M. Czarnecki, J. Schwarz, J. Rae, S. Osindero, Y. W. Teh, T. Harley, and R. Pascanu. Multiplicative interactions and where to find them. In *International Conference on Learning Representations*, 2020.

B. Lengerich, S. Tan, C.-H. Chang, G. Hooker, and R. Caruana. Purifying interaction effects with the functional anova: An efficient algorithm for recovering identifiable additive models. In *International Conference on Artificial Intelligence and Statistics*, pages 2402–2412, 2020.

A. C. Leon and M. Heo. Sample sizes required to detect interactions between two binary fixed-effects in a mixed-effects linear regression model. *Computational statistics & data analysis*, 53(3):603–608, 2009.

Y. Lou, R. Caruana, and J. Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158. ACM, 2012.

Y. Lou, R. Caruana, J. Gehrke, and G. Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631. ACM, 2013.

P. Mianjy, R. Arora, and R. Vidal. On the implicit bias of dropout. *arXiv preprint arXiv:1806.09777*, 2018.

P. Nakkiran, G. Kaplun, D. Kalimeris, T. Yang, B. L. Edelman, F. Zhang, and B. Barak. Sgd on neural networks learns functions of increasing complexity. *arXiv preprint arXiv:1905.11604*, 2019.

E. Nalisnick, J. M. Hernández-Lobato, and P. Smyth. Dropout as a structured shrinkage prior. *arXiv preprint arXiv:1810.04045*, 2018.

D. Nielsen. Tree boosting with xgboost-why does xgboost win" every" machine learning competition? Master's thesis, NTNU, 2016.

I. Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS Workshop on Bayesian Deep Learning*, volume 192, 2016.

S. Park and N. Kwak. Analysis on the dropout effect in convolutional neural networks. In *Asian Conference on Computer Vision*, pages 189–204. Springer, 2016.

L. Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

D. Selsam, M. Lamm, B. Bünz, P. Liang, L. de Moura, and D. L. Dill. Learning a sat solver from single-bit supervision. *arXiv preprint arXiv:1802.03685*, 2018.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

S. Tan, R. Caruana, G. Hooker, P. Koch, and A. Gordo. Learning global additive explanations for neural nets using model distillation. *arXiv preprint arXiv:1801.08640*, 2018.

J. K. Tay and R. Tibshirani. Reluctant additive modeling. *arXiv preprint arXiv:1912.01808*, 2019.

Y. Tsuzuku and I. Sato. On the structural sensitivity of deep convolutional networks to the directions of fourier basis functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 51–60, 2019.

S. Wager, S. Wang, and P. S. Liang. Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pages 351–359, 2013.

L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

M. Wand and J. T. Ormerod. Penalized wavelets: Embedding wavelets into semiparametric regression. *Electronic Journal of Statistics*, 5:1654–1717, 2011.

H. Wang, X. Wu, Z. Huang, and E. P. Xing. High-frequency component helps explain the generalization of convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8684–8694, 2020.

R. Wang, B. Fu, G. Fu, and M. Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, page 12. ACM, 2017.

D. Warde-Farley, I. J. Goodfellow, A. Courville, and Y. Bengio. An empirical analysis of dropout in piecewise linear networks. *arXiv preprint arXiv:1312.6197*, 2013.

A. Weigend. On overfitting and the effective number of hidden units. In *Proceedings of the 1993 connectionist models summer school*, volume 1, pages 335–342, 1994.

B. M. Wilamowski, D. Hunter, and A. Malinowski. Solving parity-n problems with feedforward neural networks. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 4, pages 2546–2551. IEEE, 2003.

D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer. A fourier perspective on model robustness in computer vision. In *Advances in Neural Information Processing Systems*, pages 13255–13265, 2019.

A. Zunino, S. A. Bargal, P. Morerio, J. Zhang, S. Sclaroff, and V. Murino. Excitation dropout: Encouraging plasticity in deep neural networks. *arXiv preprint arXiv:1805.09092*, 2018.

Supplementary Material:
Dropout as a Regularizer of Interaction Effects

A   Interaction Effects

An example of the distribution changing the meaning of a pure interaction effect is shown in Fig. A.1.

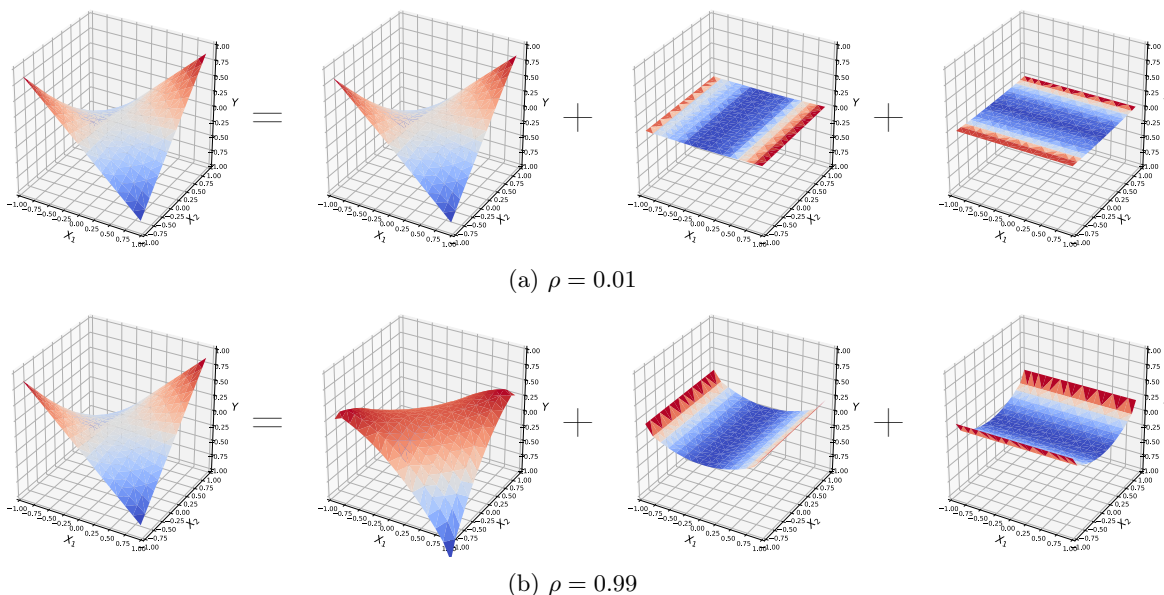(a) $\rho = 0.01$

(b) $\rho = 0.99$

Figure A.1: A toy example of decomposing a function into pure interaction and main effects. In each (a) and (b), there are four panes: (left) an overall function, (middle left) a pure interaction effect of X_1 and X_2, (middle right) a pure effect of X_1, and (right) a pure effect of X_2. In both a and b, the overall function is $Y = X_1 X_2$, but the decomposition changes based on the coefficient ρ of correlation between X_1 and X_2. For X_1 and X_2 uncorrelated, the multiplication is a pure interaction effect; for X_1 and X_2 correlated, much of the variance can be moved into effects of the individual variables. The decomposition is unique given the joint distribution of the three variables.

A.1   The Unreasonable Effectiveness of Models with Few Interaction Effects

Generalized additive models (GAMs) (Hastie and Tibshirani, 1990) are a restrictive model class which estimate functions of individual features, i.e., functions of the form $f(X_i, \ldots, X_p) = \sum_{i=1}^{p} g_i(X_i)$. There have been a large number of methods for estimating these functions, including functional forms such as splines, trees, wavelets, etc. (Eilers and Marx, 1996; Lou et al., 2012; Wand and Ormerod, 2011). While vanilla GAMs describe nonlinear relationships between each feature and the label, interactions are sometimes added to further capture relationships between multiple features and the label (Coull et al., 2001; Lou et al., 2013; Tay and Tibshirani, 2019).

In the age of deep learning, it is surprising that GAMs with a small number of added interaction effects could be state-of-the-art on any dataset with a moderately large number of samples. However, successful tree-based ensembles such as XGBoost (Chen and Guestrin, 2016) often require only a few interaction effects to win competitions (Nielsen, 2016). In certain cases, polynomial regression of order 2 can be competitive with fully-connected deep NNs (Cheng et al., 2018), and even generalized additive models have a surprising capability to approximate deep NNs (Tan et al., 2018). Similar phenomena have been observed for Gaussian Processes (Delbridge et al., 2019) and computer vision models (Yin et al., 2019; Wang et al., 2020; Tsuzuku and Sato, 2019).

How are these models, which ignore the majority of interaction effects, so effective?

A.2 Statistical (Un)Reliability of Interaction Effects

One reason why models which ignore high-order interaction effects can perform so well is the tremendous difficulty that higher-order interaction effects present to learning algorithms. When trying to learn high-order interaction effects, we are stuck between a rock and a hard place: the number of possible interaction effects grows exponentially (the number of k-order interaction effects possible from N input features is $\binom{N}{k}$), while the the variance of an interaction effect grows with the interaction order (Leon and Heo, 2009). This quandry is intensified when the effect strength decreases with interaction order, which is reasonable for real data (Gelman, 2018). It is like searching for a needle in a haystack, but as we increase k, the haystack gets larger and the needle gets smaller. For large k, we are increasingly likely to select spurious effects rather than the true effect – at some point it is better to stop searching the haystack. Viewed this way, it is less surprising that in the absence of prior knowledge of which interaction effects are true, simple models are able to outperform large models.

A.3 Parity and Interaction Effects

Interaction effects are intricately linked to a classically difficult function class: parity. In the case of two Boolean variables, a pure interaction effect is exactly a weighted XOR function and for continuous variables, pure interaction effects are a continuous analog of parity (Lengerich et al., 2020). Parity functions are notoriously difficult to learn with NNs (Wilamowski et al., 2003; Selsam et al., 2018). Does this suggest that NNs are already robust against interaction effects, and if so, why is the extra regularization of Dropout against interaction effects necessary?

It is important for us to distinguish between learning the *correct* interaction effect against learning a *spurious* interaction. Given N variables, there are $O(N)$ possible main effects, $O(N^2)$ possible pairwise interactions, $O(N^3)$ possible 3-way interactions, $O(N^4)$ possible 4-way interactions, etc. This exponential growth in the hypothesis space of interaction terms simultaneously increases the probability that a universal approximator would estimate *some* interaction effect while decreasing the probability that the same universal approximator selects the *correct* interaction effect. For this reason, it can be possible for model classes to struggle with accurate recovery of parity functions without being inherently biased against high-order interactions. As shown in Figure 3, the exponential growth in the number of potential interaction terms is balanced by the exponential decay in learning rate induced by Dropout. In this way, large NNs trained with Dropout can have the convenient property that they are *capable* of learning high-order interactions but will put off the difficult task of learning these high-order interactions until simpler functions have been thoroughly explored.

B Analysis

B.1 Proof of Theorem 1

Proof. Let $\mathbb{E}[Y|X] = F(X) = \sum_{u \in [d]} f_u(X_u)$ and $\tilde{Y} = F(X \odot M_p)$, where $M_p \sim Bernoulli(p)$ is the Input Dropout mask and $\odot$ is element-wise multiplication. Then

$$\mathbb{E}_M[\tilde{Y}|X] = \sum_{u \in [d]} P(X \odot M = X) f_u(X_u) + \big(1 - P(X \odot M = X)\big) \mathbb{E}_M[f_u(X_u \odot M_u^+)] \tag{7a}$$

$$= \sum_{u \in [d]} (1-p)^{|u|} f_u(X_u) + \big(1 - (1-p)^{|u|}\big) \mathbb{E}_M[f_u(X_u \odot M_u^+)] \tag{7b}$$

$$= \sum_{u \in [d]} (1-p)^{|u|} f_u(X_u) + \big(1 - (1-p)^{|u|}\big) \mathbb{E}_{v \in u}[f_u(X_{u \setminus v}, X_v = 0)] \tag{7c}$$

$$= \sum_{u \in [d]} (1-p)^{|u|} f_u(X_u) + a_u(X_u) \tag{7d}$$

where M^+ is drawn uniformly from the Dropout masks with at least one zero value and $a_u(X_u) = \sum_{v \subseteq u} p^{|u|-|v|}(1-p)^{|v|} f_U(X_{u \setminus v}, X_v = 0)$. Further,

$$\mathbb{E}_{X_u}[a_u(X_u)] = \mathbb{E}_{X_u}[\sum_{v \subseteq u} p^{|u|-|v|}(1-p)^{|v|} f_U(X_{u \setminus v}, X_v = 0)] \tag{8a}$$

$$= \sum_{v \subseteq u} p^{|u|-|v|}(1-p)^{|v|} \mathbb{E}_{X_u}[f_u(X_{u \setminus v}, X_v = 0)] \tag{8b}$$

$$= 0 \tag{8c}$$

where the final equality holds by orthogonality of the fANOVA. $\qquad\square$

B.2 Proof of Corollary 1.1

Proof. Let $f_u(X_u)$ be a multiplicative effect. Then

$$f_u(X_{u \setminus v}, X_v = 0) = 0 \quad \forall \, v \, \in \, u \tag{9}$$

and hence $a_u(X_u) = 0 \, \forall \, u$. $\qquad\square$

B.3 Proof of Corollary 1.2

Proof. Let $F(X) = G(X, Y, \theta)$ for fixed Y, θ. Then $\mathbb{E}[F(X \odot M_p)] = \sum_{u \in [d]} (1-p)^{|u|} f_u(X) + a_u(X_u)$ by Theorem 1. $\qquad\square$

B.4 Proof of Corollary 1.3

Proof. Let $F(A_i) = G_i(A_i, Y, \theta)$ for fixed Y, θ. Then $\mathbb{E}[F(A_i \odot M_p)] = \sum_{u \in [d]} (1-p)^{|u|} f_u(X) + a_u(X_u)$ by Theorem 1. $\qquad\square$

C Additional Experimental Details

Pure Noise Data   Figure C.1 shows the results of various Dropout rates on a NN with 128 hidden units in each layer. These results are analogous to the results shown in Fig. 5 of the main text for a NN with 32 hidden units in each layer.

Modified 20-NewsGroups   Table C.1 displays the results of various Dropout Rates on the Modified 20-NewsGroups datasets described in Section 4.3.

| k	Dropout Rate					
	0.0	0.125	0.25	0.375	0.5	0.625
1	0.52 ± 0.01	0.54 ± 0.01	0.54 ± 0.03	$\mathbf{0.57 \pm 0.02}$	0.55 ± 0.02	0.47 ± 0.02
2	0.39 ± 0.01	0.38 ± 0.03	$\mathbf{0.40 \pm 0.02}$	$\mathbf{0.40 \pm 0.01}$	0.38 ± 0.01	0.27 ± 0.02
3	0.39 ± 0.01	$\mathbf{0.41 \pm 0.01}$	$\mathbf{0.41 \pm 0.01}$	0.40 ± 0.02	0.40 ± 0.02	0.27 ± 0.04

Table C.1: Test accuracies of the models trained on the modified 20-Newgroups datasets (Sec. 4.3). Reported values are (mean $\pm$ std) of the test accuracies over 5 experiments, with the best setting in each row bolded. Each row indicates k, the order of the added interaction effect. As k is increased, lower levels of Dropout tend to outperform. Different modifications of the dataset change the difficulty of the task, so the accuracy values are not comparable across rows.
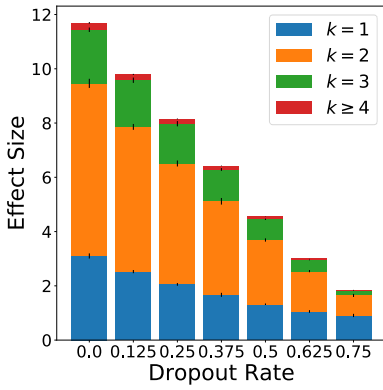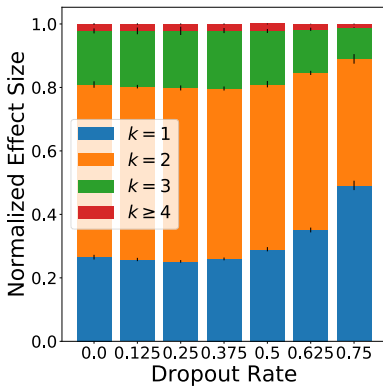
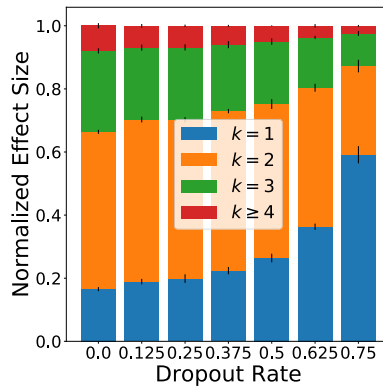(a) Total Effect: Activation Dropout
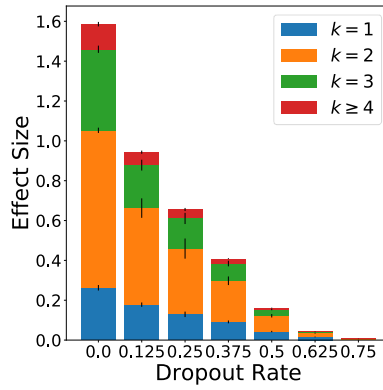
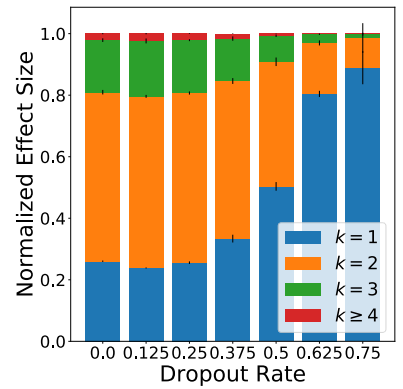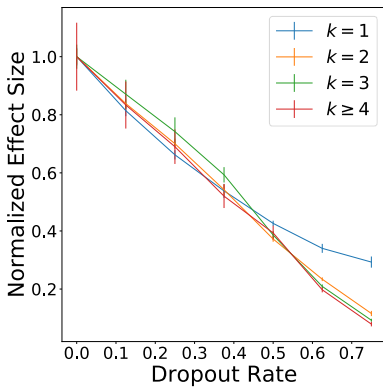(b) Total Effect: Input Dropout
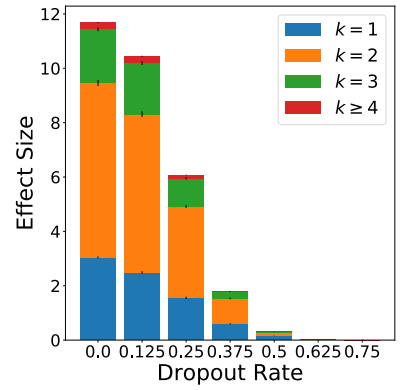
(c) Total Effect: Input + Activation Dropout
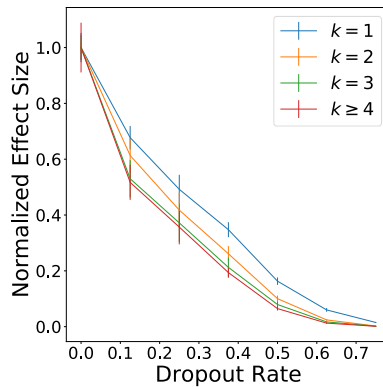
(d) Normalized Effect: Activation Dropout

(e) Normalized Effect: Input Dropout
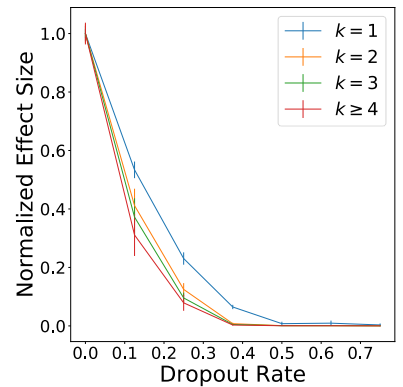
(f) Normalized Effect: Input + Activation

(g) Shrinkage: Activation Dropout

(h) Shrinkage: Input Dropout

(i) Shrinkage: Input + Activation Dropout

Figure C.1: In this experiment, we train fully-connected neural networks on a dataset of pure noise (details in Sec. 4.2). Displayed values are the (mean ± std. over 10 initializations) of the proportion of the trained model's variance explained by each order of interaction effect. All neural networks in this figure have 128 units in each hidden layer (compared to 32 units per layer in Figure 5), and we see that Activation Dropout has only a small impact, while Input Dropout significantly reduces the estimated effect sizes of the high-order interactions. As expected, increasing the size of the hidden layers from 32 in Figure 5 to 128 in this Figure decreases the impact of Activation Dropout on high-order interactions, but does not reduce the effectiveness of Input Dropout.