PowerGraph: Using neural networks and principal components to determine multivariate statistical power trade-offs

Ajinkya Mulay [1] **Sean Lane** [2] **Erin Hennes** [2]

Abstract

Statistical power estimation for studies with multiple model parameters is inherently a multivariate problem. Power for individual parameters of interest cannot be reliably estimated univariately since correlation and variance explained relative to one parameter will impact the power for another parameter, all usual univariate considerations being equal. Explicit solutions in such cases, especially for models with many parameters, are either impractical or impossible to solve, leaving researchers to the prevailing method of simulating power. However, the point estimates for a vector of model parameters are uncertain, and the impact of inaccuracy is unknown. In such cases, sensitivity analysis is recommended such that multiple combinations of possible observable parameter vectors are simulated to understand power trade-offs. A limitation to this approach is that it is computationally expensive to generate sufficient sensitivity combinations to accurately map the power trade-off function in increasingly high-dimensional spaces for the models that social scientists estimate. This paper explores the efficient estimation and graphing of statistical power for a study over varying model parameter combinations. We propose a simple and generalizable machine learning inspired solution to cut the computational cost to less than 10% of the brute force method while providing F1 scores above 90%. We further motivate the impact of transfer learning in learning power manifolds across varying distributions.

1. Introduction

Statistical power is quantitatively equal to the probability of rejecting a null hypothesis. Thus, it plays a crucial role in finding an effect of hypothesized interest in any study. Historically, the simplest and most direct way to improve power is to increase the study's sample size, as mathematically, it has the most significant impact concerning what the researcher can control (Cohen, 1992). However, there are practical considerations to make when increasing the sample size. In the case of a rare disease study, the total population itself might be small, difficult to locate, and even more challenging to engage, as in early- and late-stage neuro-genetic syndromes (Button et al., 2013; Szucs & Ioannidis, 2017). On the other hand, there could be access or funding issues with increasing the number of participants a researcher wishes to obtain.

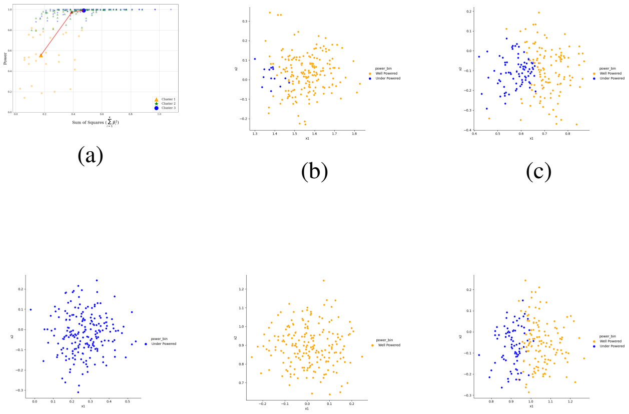
In this article, we empirically show that tuning the model parameters (*i.e.*, weights) can dramatically change the study's power. Thus, the change in model parameters can push the survey from a mid-powered study to well-powered research even for a constant sample size. For the rest of the article, we assume that well-powered research is one where the power is higher than *0.8*. Fig. 1 demonstrates that even for a fixed sample size, the power can considerab1y vary due to a change in model parameters. In Fig. 1a we compute power *gradients* by considering the directional difference between clusters derived by KMeans, where the power and the squared L2-norm of the model parameters represent the point coordinates.

Thus, generating such a manifold, as shown in Fig. 1a can exceptionally aid researchers in identifying areas of high power. Enabling access to these plots is thus a simple way of reducing the *ideal* sample size without sacrificing power. However, as described in Algorithm 1, computing power even once requires *200-1000* simulations. Further, the manifold parameter space, including the model weights, sample size, and additional hyperparameter choices, is enormous. For instance, for a seven predictor model, with 11 choices per predictor and 11 choices for the N, we have a parameter space of 11^{7+1}. Computation for this vast space might even take days.

Thus, in this work, we look at *cheaper* alternatives to simulating the entire power manifold. We present three different contributions

- We develop a neural network-based approach to predict power over a high-dimensional manifold for classification and regression purposes. Such an approach provides significant performance with even 10% of the training data.

- We provide four baselines for alternatively predicting power and test all our approaches on three models-= regression, logistic and Repeated Measures ANOVA.
- To leverage the transfer of knowledge from one model's power manifold to another model's manifold, we first demonstrate the intuition behind such a transfer. Next, we show that such a transfer can enable the same or better results in every case. We organize the rest of the article as follows. We first detail the relevant literature and highlight the background information required to build our algorithm. We provide pseudocodes for the core and auxiliary algorithms in this section. Next, we present our empirical results, highlight the significant wins, and report the limitations. Finally, we also present potential future work.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 1: **Impact of Model Parameters on Power:** *(a)* Change in power over varying model coefficients (β), $N = 105$. We observe significant power gradient over clusters derived by KMeans for a 3-predictor regression model with a partial f-test. *(b, c, d, e, f)* For the same partial f-test and over popular regression datasets (KumarRajarshi, 2015; Eso, 2019; Choi, 2018; Too, 2019; Par, 2022) ($N = 50$), we demonstrate that if we perturb the estimated model parameters ($\hat{\beta}$) by even small amounts, the under-powered studies can become well-powered (*i.e.,* power > 0.8).

2. Related Work

Previous articles ((Bakker et al., 2012), (Bakker et al., 2016), (Maxwell, 2004), (Cohen, 1992)) have pointed out that studies are frequently underpowered and thus lead to statistically insignificant results. Underpowered studies are primarily a result of a lack of a formal power analysis. Furthermore, in (Card et al., 2020) the authors demonstrated that numerous NLP models are typically underpowered. They mention that datasets with sentences less than 2000 lead to a power of only 75% and provide best practices to perform power analyses. Next, the work in (Koehn, 2004) provides ways to measure the statistical significance of the performance improvement due to a system change in the domain of machine translation. The authors argue that just a change in the system metric (ex., accuracy or F1-score) is not enough, and we need to demonstrate that the change in the metric is statistically significant. This work further broadens the need for thorough power analysis. Finally, (Koehn, 2004) takes into consideration the effects of model complexity in extracting the maximum information from a given dataset. Under a given class of models, the authors provide an algorithm to identify the *right-fit* model. The right-fit model avoids both underfitting and overfitting and has predictive power asymptotically close to the *best* model in the given model class. Such a method can replace cross-validation and signifies the importance of improving predictive power.

(Bakker et al., 2012) suggested power simulations for studies using Questionable Research Practices (QRPs) (John et al., 2012) and lower sample sizes with more trials. QRPs could include running multiple tests with a much smaller sample size (underpowered), rerunning analyses after adding more subjects, or (even selectively) removing outliers. The results show that such practices can significantly inflate statistical power and provide misleading evidence about the effect size while hampering the study's reproducibility. (Bakker et al., 2012) suggests that to avoid such underpowered studies, we should use sample sizes derived after a formal power analysis.

(Baker et al., 2021) solved the power issue by providing an online tool with power contours to demonstrate the effect of sample size (N) and trials per participant on statistical power (k). With the results provided in the article, it is clear that changes to N or even k can dramatically change the power region and convert an underpowered study to an appropriately powered one. (Rast & Hofer, 2014) also demonstrates the strong (inversely correlated) impact of sample size on both the effect size and study design.

(Lane & Hennes, 2018) and (Lane & Hennes, 2019) provide a clear guide to conducting formal power analysis based on simulation methods (rather than a formula-based approach). Even though the simulation approach is universal, it often requires many computational resources. The resource usage increases exponentially with a linear increase in the number of predictors or other factors impacting power. Thus, in this work, we highlight machine learning-based approaches that can seriously reduce resource usage with accuracies greater than 95%.

3. Proposed Work

3.1. Computing Power

We present a power computation algorithm in Algorithm 1 for t-tests in a linear regression model. We can extend this algorithm to the f-test by replacing the t-test in Line 9 with an f-test or a partial f-test. For extensions to other models, we only need to modify the data generation process on *Line 5.* Computing power for multiple model weights requires us to call the *COMPUTE-POWER* function each time. Thus, the cost of computing power boils down to the number of calls made to *COMPUTE-POWER*. We wish to reduce the number of calls while still predicting power for the entire parameter space in our work.

3.2. Feature Engineering with PCA

Complex datasets include several features, and it often becomes necessary to reduce data dimensionality to conserve resources or speed up training. Furthermore, we wish to remove redundancy in features. Removing highly correlated features can improve training speeds or data processing speeds, reduce bias, and improve the interpretability of our dataset. A common way to achieve these goals is to use Principal Component Analysis (PCA) (Wold et al., 1987). For brevity, we add further details of PCA to the appendix.

We already know that increasing the sample size N can increase the power of a study. To visualize the impact of each parameter in the model, we take a look at a partial f-test. The data follows the distribution of the first three variables from Table 5. We compute the power of the partial f-test to test for the significance of the first and third predictors.

To visualize the importance of each parameter, we draw a correlation plot in Figure 13. We can easily verify that as expected β_1 and β_3 have higher correlation with the power (as the hypothesis depends on them). N has a high correlation as we would expect. Further, we define a new feature *scaled weight* computed as $N\sigma$, wherein $\sigma = \sqrt{\sum_{i=1}^{k} \beta_i^2}$ (k denotes the number of predictors). We observe that the $N\sigma$ feature has the highest correlation amongst non-PCA features. After including the principal components the correlation substantially increases to ≥ 0.90 for PC$_1$. We believe that transforming the data with PCA can further increase the variance while ignoring the redundant features in our dataset. We, provide exact details about our dataset in 4.2.

3.3. Primary Approach: The Neural Network Approach

We employ a simple neural network as described in Table 4 trained on the *new* dataset. With the flexibility of a neural network, we can train for both a classification and a regression task. We denote this approach by *POWER-NETWORK* or *POWER-Neural Network* (*PNN*)). *PNN* also allows for an easy extension to a multi-class classifier. However, we do not explore this domain in this article.

3.3.1. Learning Faster with Transfer Learning

The primary goal of our article is to reduce the number of calls to the power function and thus reduce the simulation overhead for computing power. Thus, we now describe a commonly used technique *fine-tuning* (Houlsby et al., 2019) in Natural Language Processing (NLP) domains. Rather than cold-starting model training with random weights, we take the help of a previously trained model to initialize the model weights. Thus, most layers are already trained, and our new *smaller* dataset can be used to tune the neural net for our new task. Note that when the number of input dimensions is different we remove the unused dimensions (from the pre-trained model) by specifying zero input columns for those features.

Formally, the transfer learning algorithm *P-Transfer* only deviates from *PNN* in the fact that line 6 of the algorithm 4 has an additional step of initializing the model (before training) with the weights of a larger pre-trained model. Note that the original pre-trained model needs more features than the new model. Finally, the extra features in the larger model are turned off by passing zero vectors to these features. Thus, even a larger model can be used for our smaller models.

3.3.2. Intuition behind Transfer Learning

The intuition behind transfer learning may be sought from this article (Chinn, 2000). We first note that each model directly interacts with the linear model. However, due to the difference in their composition, the effect size might differ. The authors in(Chinn, 2000) empirically demonstrate that given the standard deviation, we can find the correlation between, say, the effect size of the logistic regression model and the regression model. Thus, we would have the same standard deviation and thus correlated/comparable effect sizes for standardized predictors. Thus, we might be able to use transfer learning to boost a model with similar traits as long as the predictors are standardized and they both have the same hypothesis.

3.4. Baselines: Power Cluster, Power Label Propagation and KNeighbors Classifier

POWERCLUSTER also referred to as *P-CLUSTER*) simply runs K-Means on PC_1 and $N\sigma$ for identifying two clusters with two vectors from our dataset. We provide its pseudocode in Algorithm 2. K-Means identifies clusters of points that minimize the intra-cluster distances. Since both of our feature vectors are independent of power, we can cluster our data points without computing the *true* power.

The intuition behind deploying *P-CLUSTER* is that it can quickly identify appropriately varying power domains, as seen from Fig. 2. If we compare the clustering to the standard definition of a high-powered study (*i.e.,* power is more significant than *0.8*), we can segregate power with high performance. We demonstrate the results in Figures 3a, 3b. We can see that *P-CLUSTER* performs well arbitrarily. Thus, we can conclude that not including label information will lead to poor performance for models.

We provide two other strong baselines for power prediction- the label propagation algorithm from (Zhu & Ghahramani, 2002), and the KNeighbors Classifier (Nelli, 2018). We use the Label Propagation version implemented in Scikit-Learn (Pedregosa et al., 2011). Label propagation works by first creating a fully-connected graph between all data points wherein each edge weight is directly dependent on the euclidean distance between the two points. These edge weights dictate the probability of propagating a label onto another label. We direct the reader to section 2.2 of (Zhu & Ghahramani, 2002) for further details on the Label Propagation algorithm. We also refer to it as *PL-PROP*.

KNeighbors classifier (Nelli, 2018) also provides good performance, and we thus include it for completeness. Overall, Label Propagation performs consistently better than the KNeighbors classifier.

4. Experiments

4.1. Code

We test the efficacy of the baseline algorithms and 4 with three kinds of statistical models. For simulation, we use Google Colaboratory with the standard run-time.[1] Other experiment details are deferred to the appendix in section B.1.1.

Algorithm 1 Computing Power of a t-test for a Linear Regression Model

```
Input: Distributions of columns in the dataset-$\mathcal{D} = \{D_{X_1}, D_{X_2}, ..., D_{X_p}\}$, Sample Size-$N$, Model Weight-$\beta \in \mathbb{R}^p$, Number of predictors-$p$, sensitivity-$\alpha$ (default: 0.05), number of simulations-sims (default: 1000), Error Distribution-$\mathcal{E}$
Output: Power of t-test
procedure COMPUTE-POWER($X, k$)
    significance $\leftarrow 0$
    for $1$:sims do
        $X \leftarrow$ generate $N$ data samples from
        distribution($\mathcal{D}$)
        $e \leftarrow$ generate error from distribution($\mathcal{E}$)
        $y \leftarrow (X \times \beta) + e$
        $\mathcal{M} \leftarrow$ Fit $(X, y)$ to a linear regression model
        if p-value of t-test for model $\mathcal{M} \leq \alpha$ then
            significance += 1
        end if
    end for
    power $\leftarrow \frac{significance}{sims}$
    return power
end procedure
```

Algorithm 2 Unsupervised clustering of the power surface with PCA features

```
Input: Length of training set-$L$, model parameter space-$\mathcal{S}$ of length $L$ (each parameter consists of the weight vector $\beta$ and sample size $N$), power sensitivity-$\alpha$ (default 0.05), number of simulations-sims (default 200), PCA variance (in %) to be retained-var, Number of clusters-$k$
Output: Power Surface Graph
procedure POWERCLUSTER($\mathcal{S}$, $\alpha$, sims)
    $\mathcal{S}_{\text{PCA}} \leftarrow$ PCA-FIT-TRANSFORM(S, variance=var)
    $C_k \leftarrow$ KMeans($\mathcal{S}_{\text{PCA}}$, num_clusters = $k$)
    return clusters $C_k$ derived from KMeans
end procedure
```

4.2. Dataset

For sampling parameters we use *P-SAMPLER* (algorithm 5) to generate the parameter sample space and in conjunction with algorithm 3 to compute power. Both of these algorithms together make up the power computation black box. After collecting the dataset, we pre-process it using PCA, as explained below. *P-SAMPLER* is inspired by the fungible weights described in (Waller, 2008). Next, we provide exact details about the data we use for the methods introduced in this work. For both supervised and unsupervised tasks we use the same dataset albeit the unsupervised one does not contain the labels or true power information. The dataset

[1] Our code is available at the anonymized link `https://anonymous.4open.science/r/powergraph-4BFC`.

Algorithm 3 Training Data Collection

```
Input: Length of training set-$L$, model parameter space-$\mathcal{S}$ of length $L$ (each parameter consists of the weight vector $\beta$ and sample size $N$), power sensitivity-$\alpha$ (default 0.05), number of simulations-sims (default 200)
Output: Training Set $\mathcal{S}$ with parameters and corresponding powers
procedure GENERATE-DATA($\mathcal{S}$, $\alpha$, sims)
    for parameter in $\mathcal{S}$ do
        $(\beta, N) \leftarrow$ parameter
        power $\leftarrow$ COMPUTE-POWER($\beta, N, \alpha$,sims)
        $\mathcal{S}$[powers] $\leftarrow$ power
    end for
end procedure
```

Algorithm 4 Classifying/Predicting power surface with PCA features

```
Input: Length of training set-$L$, model parameter space-$\mathcal{S}$ of length $L$ (each parameter consists of the weight vector $\beta$ and sample size $N$), power sensitivity-$\alpha$ (default 0.05), number of simulations-sims (default 200), PCA variance (in %) to be retained-var
Output: Power Surface Graph
procedure POWERNETWORK($\mathcal{S}$, $\alpha$, sims)
    $\mathcal{S}^* \leftarrow$ GENERATE-DATA($\mathcal{S}$, $\alpha$, sims)
    $\mathcal{S}_{\text{PCA}} \leftarrow$ PCA-FIT-TRANSFORM($\mathcal{S}$, variance=var)
    Train Neural Network Model $\mathcal{M}$ on ($\mathcal{S}^* \cup \mathcal{S}_{\text{PCA}}$)
    return trained model $\mathcal{M}$
end procedure
```

Algorithm 5 Sampling in High-Dimensions

```
Input: no. of training points-$N_s$, domain of the $k$ predictors and the sample size-$\mathcal{D} \in \mathbb{R}^k$, no. of local training points-$n_s$, local sigma for Gaussian sampling-$\sigma_l$
Output: $\mathcal{S}$ model parameter space
procedure P-SAMPLER($N_s$)
    $n \leftarrow 0$
    while $n < N_s$ do
        Uniformly randomly pick a centroid $\beta$ from $\mathcal{D}$
        $\mathcal{S}_l \leftarrow$ Sample $n_s$ datapoints from a Gaussian distribution $\mathcal{N}(\mu = \beta, \sigma = \sigma_l)$
        $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_l$
        $n \leftarrow n + 1 + n_s$
    end whilereturn parameter space $\mathcal{S}$
end procedure
```

details are included in Table 1. We assume that our original linear model is denoted as

$$y = x_1\beta_1 + x_2\beta_2 + ... + x_k\beta_k + e$$

wherein, $e \sim \mathcal{N}(0, 1)$ is the error term, k is the number of predictors, y is the model predictions, $x = \{x_1, ..., x_k\}$ and $\beta = \{\beta_1, ..., \beta_k\}$ is our linear model parameter. Our modified dataset includes $X = \{\beta, N, N\sigma\}$. After including the PCA induced features our final dataset is $X_{PCA} = \{\beta, N, N\sigma, \text{PC}\}$.

4.2.1. Models

Three models are considered for understanding how the power manifold is impacted by model complexity. x below signifies the number of predictors in the model.

- *Linear Model (REG-x):* We run an f-test to understand whether a simpler model works better than the full model. Feature distribution listed in Table 5.

- *Logistic Regression (LOGIT-x):* Here, we run a Wald's test to understand how much a reduced model impacts the predictive power. This is the only non-linear model that we consider. Feature distribution listed in Table 6.

- *Repeated Measures ANOVA (RMANOVA-x):* We run a within-subjects RMANOVA to measure the impact of different factors (two) on the subjects. Feature distribution listed in Table 5.

Throughout the paper, we use the same hypothesis $\mathcal{H}_{\mathcal{O}}$ where we test whether the model parameters β_1, β_3 are zero (in the form of a partial F-test or a Wald's test). However, for testing the robustness of our transfer learning algorithm *P-TRANSFER*, we use the alternative hypothesis $\mathcal{H}_{\mathcal{O}}^{'}$ where we test whether the model parameters $\beta_1, \beta_7, \beta_8$ are zero or not for both an F-test and a Wald's test.

Feature	Details
β	As provided by the expected dataset's distribution
$N, N\sigma$	Sample Size, Scaled Weight
PC = $\{\text{PC}_1, ..., \text{PC}_r\}$	r Principal Components as extracted from X with 99% variance

Table 1: Dataset: Features included for model training

For the *3*-predictor network we have *9* total features, *13* features for the *5*-predictor network, *23* for the *10*-predictor network and *43* for the *20*-predictor network.

4.3. Metrics

In our proposed algorithm, we solve two kinds of problems-binary power classification and a regression problem with actual power values.

4.3.1. Classification

For binary classification, we normally simply use the he accuracy. However, since we might have unbalanced datasets, we instead report the F1 score over the test set. While using multiple binary classifiers we could still use the F1-score albeit in an *one-vs-rest* fashion. For multi-class classification, we are first required to run the more restrictive binary classifier, compute its performance, and then for the second binary classifier we only compute its performance over the remaining data points. Thus, suppose we have two binary classifiers- C_1 (*power* > 0.8 and *power* ≤ 0.8) and C_2 (*power* > 0.6 and *power* ≤ 0.6). We first run C_1 to find all data points with *power* > 0.8 and then run C_2 to find data points with $0.8 \geq$*power* > 0.6 and *power* ≤ 0.6. Note that currently in the results we only provide separate performance metrics for C_1 and C_2.

4.3.2. Regression

We use a Mean Squared Error loss function for the regression problem between the ground truth power and the simulated power values. However, power is the probability of successfully rejecting a null hypothesis, and thus we can consider the ground truth power P^g and the simulated power values P^s as probability distributions. Note that the KL Divergence between A and B is shown in equation 1.

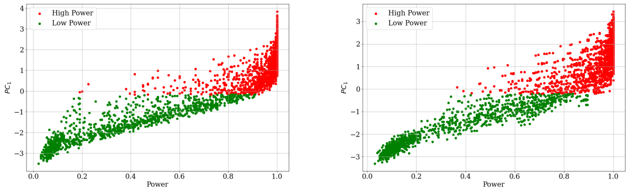
$$\mathcal{L}_{KL}(A||B) = \sum_{x} A(x)\log\Big(\frac{A(x)}{B(x)}\Big) \tag{1}$$

Here, x denotes the feature space associated with the probability distribution (*i.e.,* actual and simulated power), and A, B represents the ground truth and the simulated powers themselves. However, KL divergence is non-symmetric and not a distance measure. Thus, we use the Jensen-Shannon (JS) divergence which is an extension of the equation. 1. The JS divergence between two probability distributions A and B is given by the equation 2.

$$\mathcal{L}_{JS}(A||B) = \frac{1}{2}\{\mathcal{L}_{KL}(A||B) + \mathcal{L}_{KL}(B||A)\} \tag{2}$$

4.4. A note about current baselines for predicting power

From literature, previous approaches for computing a power manifold used a brute-force approach by essentially running the *black box* for computing power. Thus, given unlimited time, they could achieve 100% accuracy by just literally computing the power manifold. Specific greedy approaches attempted to find optimal power but not the entire manifold. Since our work aims to minimize the effort required to compute the power manifold, we do not compare our work directly to these brute-force approaches. Instead, we elect to create our baselines using traditional predictive machine learning.



(a)

(b)

Figure 2: **PNN-CLUSTER Performance:** A simple unsupervised learning algorithm can identify high and low power domains. *(left)* Results for partial f-test performed on a 3-predictor *REG* model. *(right)* Results for partial f-test performed on a 5-predictor *REG* model.

5. Results

5.1. P-Transfer

We evaluate P-Transfer by first pre-training a *REG-20* model and then testing it over three different scenarios-(i) Same feature distribution, Same Hypothesis, (ii) Different feature distribution, Same Hypothesis, and (iii) Different feature distribution, Different Hypothesis. We denote these cases by TF_1, TF_2, TF_3 respectively. Note that we always use a model smaller or equal in size to the original pre-trained model for transfer learning.

Our preliminary observations indicate that transfer learning *never* hurts model performance. However, for the cases where both the models have the same hypothesis (*i.e.,* cases (i) and (ii)), we see significant gains in the model performance. Table 2 summarizes these results.

Thus, for comparison to *PNN*, we use our baselines *-PL-Rand, P-CLUSTER, PL-PROP* and the *PK-neighbors* approach. Note that to accommodate the various trends in our model, we only report the best performing baseline (over all four baselines) in the plots. For brevity and due to many baseline performance combinations, we skip the point-wise results of the baselines. However, the code provided hosts with all the information required to recreate these baselines.

We capture three primary trends in our experiments, the performance over the change in model complexity, the number of predictors, and the amount of training data used for predicting the power manifold.

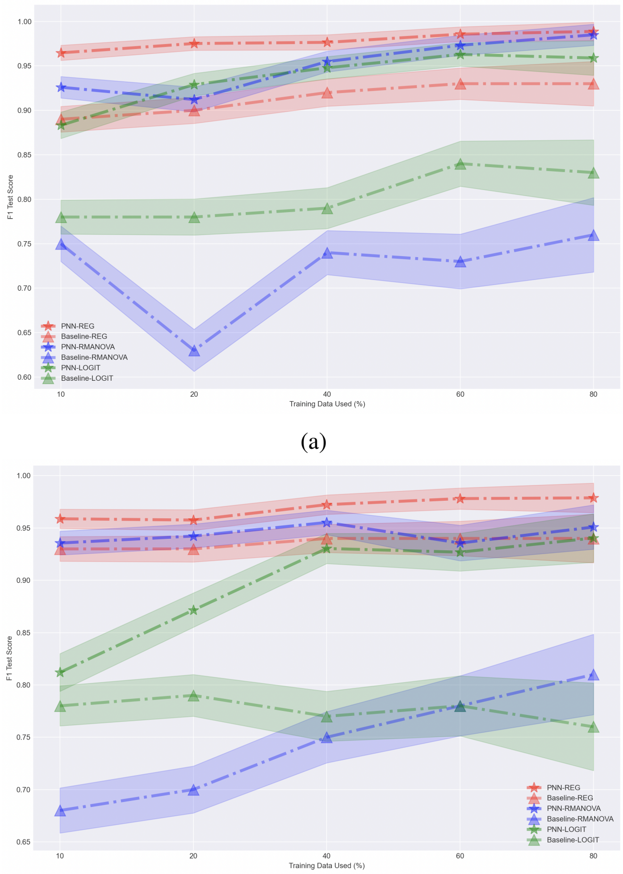
Type	Model	P-Transfer Performance	Original Performance (Same Amt. of Train Data)	Original Performance (8x Amt. of Train Data)
TF_1	REG3	$\mathbf{0.9902^{\dagger} \pm 0.0045}$	0.9827	0.9900
TF_1	REG5	$\mathbf{0.9804^{\dagger} \pm 0.0064}$	0.9700	0.9800
TF_1	REG10	$\mathbf{0.9630 \pm 0.0087}$	0.9600	0.9800
TF_1	RMANOVA3	$\mathbf{0.9458 \pm 0.0104}$	0.945	0.9600
TF_1	RMANOVA5	$\mathbf{0.9650 \pm 0.0084}$	0.9600	0.9700
TF_1	RMANOVA10	$\mathbf{0.9465 \pm 0.0103}$	0.9200	0.9800
TF_1	RMANOVA20	0.8513 ± 0.0164	**0.8600**	0.9300
TF_2	LOGIT3	$\mathbf{0.9717^{\dagger} \pm 0.0076}$	0.9300	0.9700
TF_2	LOGIT5	$\mathbf{0.9524 \pm 0.0098}$	0.9200	0.9600
TF_2	LOGIT3	$\mathbf{0.9361 \pm 0.0112}$	0.8900	0.9700
TF_2	LOGIT3	$\mathbf{0.9099 \pm 0.0132}$	0.8600	0.9300
TF_2	REG10	0.9658 ± 0.0178	**0.9659**	0.9944
TF_2	RMANOVA10	$\mathbf{0.9260 \pm 0.0256}$	0.8678	1.0000
TF_3	RMANOVA20	$\mathbf{0.815 \pm 0.0380}$	**0.815**	0.815

Table 2: *P-Transfer Test Performance*: Models with the same hypothesis may provide considerable performance boost. Any other scenario does not seem to hurt transfer learning performance.†-indicates that P-Transfer beat even the 8x training data performance. The confidence ranges indicate a 95% confidence interval.

The trend of increased performance with increased training data usage is reported via Figures 3, 4. Interestingly, even at lower training data proportions, we receive good performance from the classifiers. From the exact figures, we can see that predictions struggle over the RMANOVA model while they are much easier for the REG model and moderate for the LOGIT model. A similar trend can be observed for regression tasks as seen in the Figs. 7, 8, 9 in the appendix. Furthermore, with a change in the number of predictors as seen in Figs. 10, 11 and 12, the model performance deteriorates for change in the number of predictors for REG and LOGIT. However, for the RMANOVA model, we see an arbitrary performance plot indicating our sub-optimal model choice might be the reason. Here sub-optimal indicates the model derived without using the validation set. It seems that for both regression and classification tasks, PNN struggles the most for the RMANOVA model family. However, given that we see this is only the models with low access to training data, we can say that these classifiers are biased.

Finally, we can observe the number of calls v/s the performance in Fig. 15. Clearly, for multiple models using fewer data points can still provide high utility.

Even though we do not fully report the baselines, we would like to highlight a few notable observations. As expected, *P-RAND* performs the worst, followed by *P-CLUSTER*. Depending on the simulated data's distribution *P-CLUSTER* surprisingly beats both *PK-neighbors* and *PL-PROP* when fewer training data points are used. Amongst, *PL-PROP* and *PK-neighbors*, the *PL-PROP* approach is the more robust one and it is the closest one to *PNN*.



(a)

(b)

Figure 3: **PNN Classification Performance:** Comparison of model performance over change in proportion of training data used (increasing from left to right). **(left)** 10-predictor performance, and ***(right)*** 20-predictor network performance.

6. Limitations and Future Work

This work primarily investigates the utility of training a neural network for predicting statistical power over a manifold. Furthermore, we also provide baselines- Random (P-Rand), K Neighbors Classifier (PK-Neighbors), Label Propagation (PL-Prop), and an unsupervised clustering method (P-CLUSTER).

Note that each of the baselines fails to generalize for complex models.

- *P-RAND*: The random mechanism works as advertised and provides the least performance throughout.
- *P-CLUSTER*: This approach can be compelling for visualizing the power manifold in 2 dimensions space effectively. However, we note that its performance is inconsistent, and the linear classifier might not always

be able to predict exactly between high and low power regions. Further, the lack of label information makes this approach useful for pre-processing but not for predicting power classes. Alternatively, we might be able to identify and eliminate low powered regions using this approach, and we again propose this as an open problem.

- *PK-neighbors*: This approach is a supervised clustering alternative to *P-CLUSTER*. However, it does not necessarily perform consistently. We, however, note that we have not fine-tuned this approach for the number of neighbors. We leave this investigation as part of our future work.
- *PL-PROP*: The most promising baseline is *PL-PROP* with its *rbf* kernel. *PL-PROP* is better than the rest but fails to generalize for complex models or more predictors. Furthermore, it requires much more data compared to the neural network. We, however, have not used any guidance for tuning this algorithm's parameters, and we leave this aspect for future work.
- *Model Limitations: PNN* provides excellent empirical performance and generalizes well over increasing model complexity while providing competitive performance even with only 10% of the data. We note that *PNN* is not yet trained using methods like *early stopping* or an *adaptive learning rate*. We would pursue these optimizations in subsequent work. Particularly we do not use a validation dataset to improve the model generalization, and we set *500* epochs as an arbitrary stopping point. We have seen at least one instance of such an approach leading to a sub-optimal model choice, thus leaving a gap in the empirical performance.
- *Dataset Limitations:* Our most complex model currently is the RMANOVA with two within-subjects factors. To further understand the generalization error, we hope to extend our approach to higher complexity models, including other non-linear models. Further, we would like to explore multi-class classifications to provide the user with more options in terms of choosing the power manifold.
- *Theoretical Limitation:* Finally, the current implementations do not provide any formal theoretical guarantees for the convergence of any of these methods. We do provide intuition about why our transfer learning methods work. However, we would like to explore how different distributions with the same hypothesis impact the power manifold in greater detail.

7. Conclusion

In this work, we show that PCA-derived features are beneficial for exploring the high-dimensional manifolds of the power surface. We provide multiple algorithms that provide reasonable alternatives to the original brute force method of a power analysis. We show that using a simple, fully connected neural network, we can generalize across complex and even non-linear models to consistently predict power. We showcase that even using 10% training data can lead to high prediction accuracy for regression and classification tasks. Finally, with transfer learning, we can learn a single model and boost the performance of other models again with only 10% training data.

References

2019. URL `https://www.rdocumentation.org/packages/datasets/versions/3.6.2/topics/esoph`.

2019. URL `https://www.rdocumentation.org/packages/datasets/versions/3.6.2/topics/ToothGrowth`.

2022. URL `https://archive.ics.uci.edu/ml/datasets/parkinsons+telemonitoring`.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

Baker, D. H., Vilidaite, G., Lygo, F. A., Smith, A. K., Flack, T. R., Gouws, A. D., and Andrews, T. J. Power contours: Optimising sample size and precision in experimental psychology and human neuroscience. *Psychological Methods*, 26(3):295, 2021.

Bakker, M., Van Dijk, A., and Wicherts, J. M. The rules of the game called psychological science. *Perspectives on Psychological Science*, 7(6):543–554, 2012.

Bakker, M., Hartgerink, C. H., Wicherts, J. M., and van der Maas, H. L. Researchers' intuitions about power in psychological research. *Psychological science*, 27(8):1069–1077, 2016.

Button, K. S., Ioannidis, J. P., Mokrysz, C., Nosek, B. A., Flint, J., Robinson, E. S., and Munafò, M. R. Power failure: why small sample size undermines the reliability of neuroscience. *Nature reviews neuroscience*, 14(5): 365–376, 2013.

Card, D., Henderson, P., Khandelwal, U., Jia, R., Mahowald, K., and Jurafsky, D. With little power comes great responsibility. *arXiv preprint arXiv:2010.06595*, 2020.

Chinn, S. A simple method for converting an odds ratio to effect size for use in meta-analysis. *Statistics in medicine*, 19(22):3127–3131, 2000.

Choi, M. Medical cost personal datasets, 2018. URL `https://www.kaggle.com/mirichoi0218/insurance`.

Cohen, J. Things i have learned (so far). In *Annual Convention of the American Psychological Association, 98th, Aug, 1990, Boston, MA, US; Presented at the aforementioned conference.* American Psychological Association, 1992.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

John, L. K., Loewenstein, G., and Prelec, D. Measuring the prevalence of questionable research practices with incentives for truth telling. *Psychological science*, 23(5): 524–532, 2012.

Koehn, P. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 388–395, 2004.

KumarRajarshi. Life expectancy (who), 2015. URL `https://www.kaggle.com/kumarajarshi/life-expectancy-who?select=Life+Expectancy+Data.csv`.

Lane, S. P. and Hennes, E. P. Power struggles: Estimating sample size for multilevel relationships research. *Journal of Social and Personal Relationships*, 35(1):7–31, 2018.

Lane, S. P. and Hennes, E. P. Conducting sensitivity analyses to identify and buffer power vulnerabilities in studies examining substance use over time. *Addictive behaviors*, 94:117–123, 2019.

Maxwell, S. E. The persistence of underpowered studies in psychological research: causes, consequences, and remedies. *Psychological methods*, 9(2):147, 2004.

Nelli, F. Machine learning with scikit-learn. In *Python Data Analytics*, pp. 313–347. Springer, 2018.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Rast, P. and Hofer, S. M. Longitudinal design considerations to optimize power to detect variances and covariances among rates of change: simulation results based on actual longitudinal studies. *Psychological methods*, 19(1):133, 2014.

Szucs, D. and Ioannidis, J. P. Empirical assessment of published effect sizes and power in the recent cognitive neuroscience and psychology literature. *PLoS biology*, 15(3):e2000797, 2017.

Waller, N. G. Fungible weights in multiple regression. *Psychometrika*, 73(4):691–703, 2008.

Wold, S., Esbensen, K., and Geladi, P. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

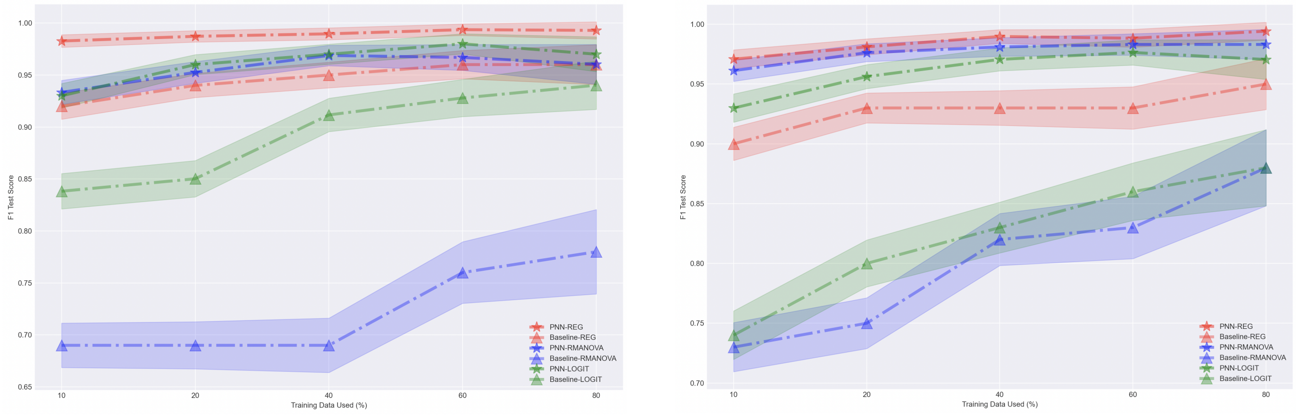
Zhu, X. and Ghahramani, Z. Learning from labeled and unlabeled data with label propagation. 2002.

A. Appendix

B. PNN Performance

We demonstrate additional results about the model's performance across changes in the number of predictors, for different classification boundaries (0.6 instead of 0.8), and the performance for regression tasks.

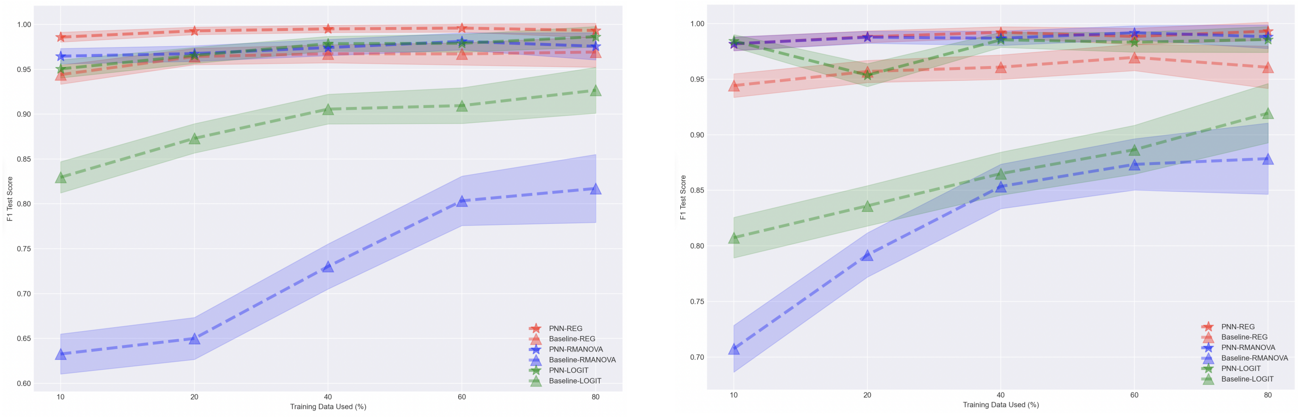
We first consider changes in the classification boundary from 0.8 to 0.6 to observe models with lower power as potentially interesting. We observe from Figs. 5 and 6.



(a)

(b)

Figure 4: **PNN Classification Performance:** Comparison of model performance over change in proportion of training data used (increasing from left to right). **(left)** 3-predictor performance, and ***(right)*** 5-predictor network performance.

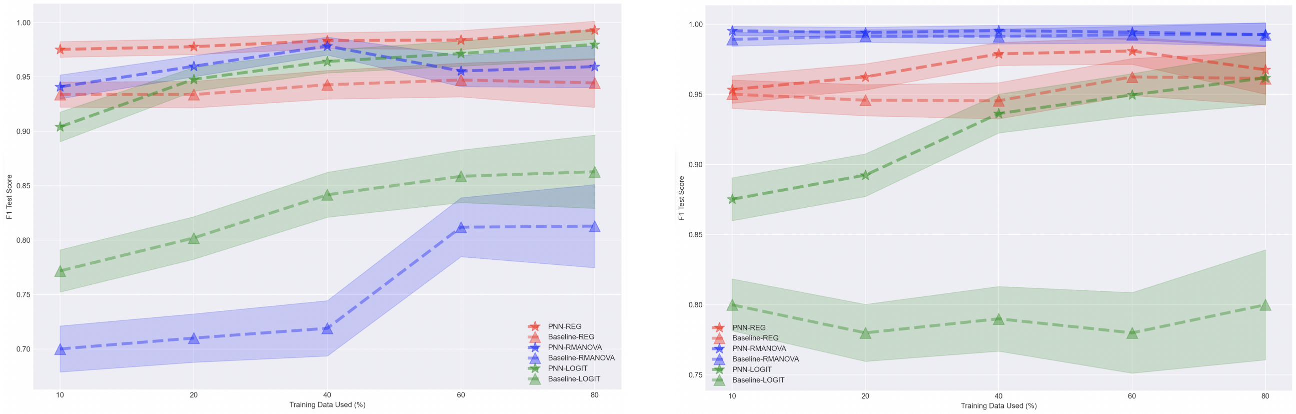


(a)

(b)

Figure 5: **PNN Classification Performance:** Comparison of model performance over change in proportion of training data used (increasing from left to right). Results for the 0.6 classification boundary for **(left)** 3 predictors , and ***(right)*** 5 predictors.

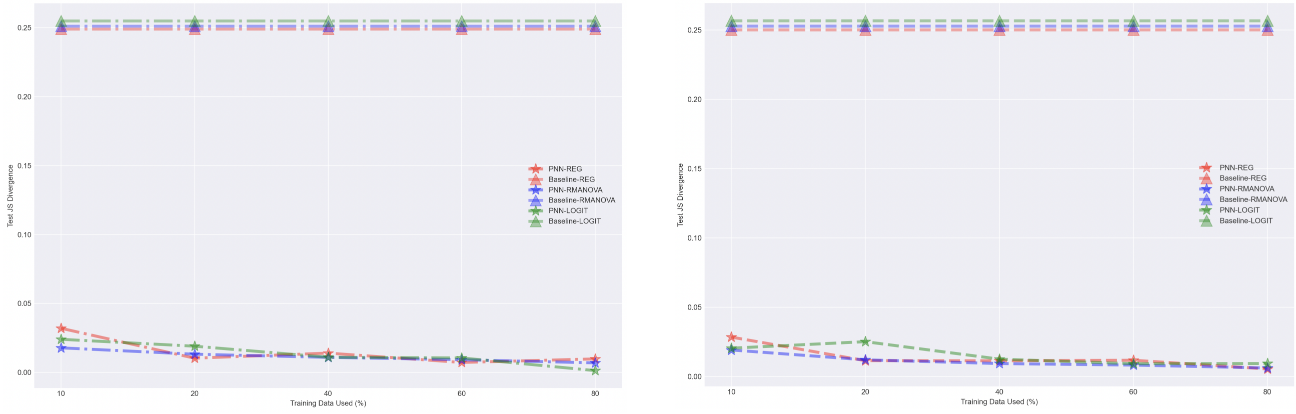
For all regression tasks in Figs. 7, 8, 9 (even over the change in the classification boundary from 0.8 to 0.6) we can observe that the JS divergence over the test set is always better than the random baseline by at least 10x and going up to even 200x performance.



(a) (b)

Figure 6: **PNN Classification Performance:** Comparison of model performance over change in proportion of training data used (increasing from left to right). Results for the 0.6 classification boundary for **(left)** 10 predictors , and ***(right)*** 20 predictors.

The performance over the change in the number of predictors as seen in Figs. 10, 11 and 12 shows a downward trend for the REG and LOGIT models and so the classifier consistently provides poor performance with an increase in model complexity due to the number of predictors. This can be explained by the massive increase in the dimensionality of the space. However, for RMANOVA we do not see such a trend at all, and rather the models can have higher performance for even the complex models. We believe this is a current limitation due to sub-optimal model choice (not driven by the validation set).

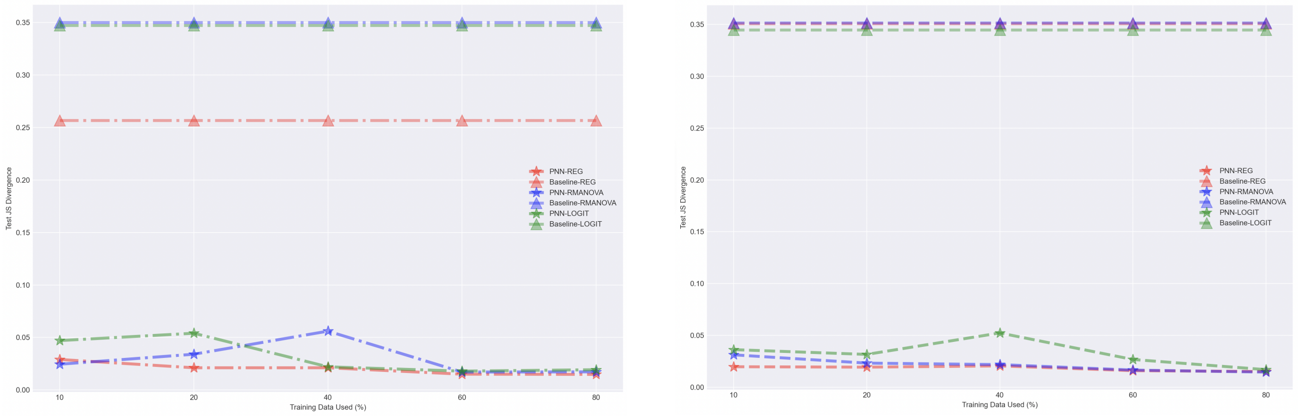


(a) (b)

Figure 7: **PNN REG Regression Performance:** Regression task is a super set of the classification task and thus inherently much harder. We report the performance of this task over change in the number of training points. Interestingly, even with just 10% of the data the neural network performs close enough to the neural network using 8x training data. Results for the REG model family ***(left)*** (classification boundary 0.8) and ***(right)*** (classification boundary 0.6).

B.1. Parameter Tuning

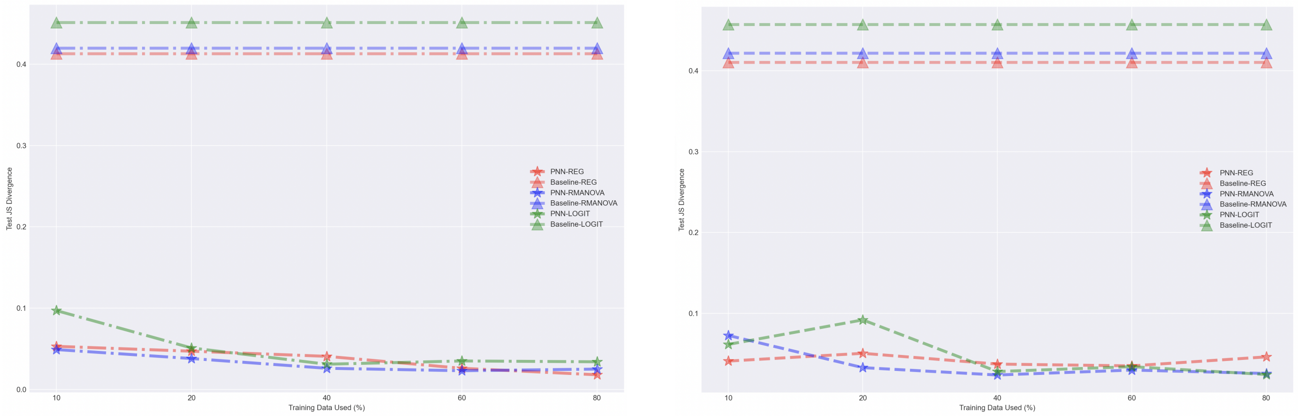
Tuning the algorithms used in this work is critical to improving the performance of the Power Network and Power Cluster. It turns out that choosing the variance captured by PCA directly affects the Power Network. Moreover, PCA variance and the number of clusters selected in Power Clustering can improve its performance.



(a)

(b)

Figure 8: **PNN RMANOVA Regression Performance:** Regression task is a super set of the classification task and thus inherently much harder. We report the performance of this task over change in the number of training points. Interestingly, even with just 10% of the data the neural network performs close enough to the neural network using 8x training data. Results for the RMANOVA model family ***(left)*** (classification boundary 0.8) and ***(right)*** (classification boundary 0.6).



(a)

(b)

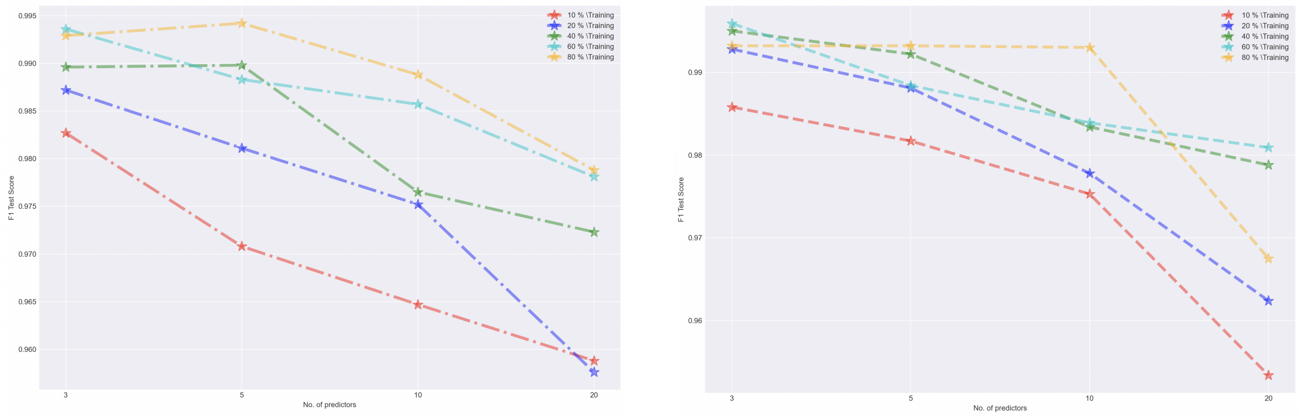
Figure 9: **PNN LOGIT Regression Performance:** Regression task is a super set of the classification task and thus inherently much harder. We report the performance of this task over change in the number of training points. Interestingly, even with just 10% of the data the neural network performs close enough to the neural network using 8x training data. Results for the LOGIT model family ***(left)*** (classification boundary 0.8) and ***(right)*** (classification boundary 0.6).

B.1.1. Experiment Details

We have standardized the parameter distribution for data collection to provide a consistent comparison. These parameters can be found in Table 3. For tuning the neural network's learning rate, we use the library Optuna ((Akiba et al., 2019)).

B.1.2. PCA

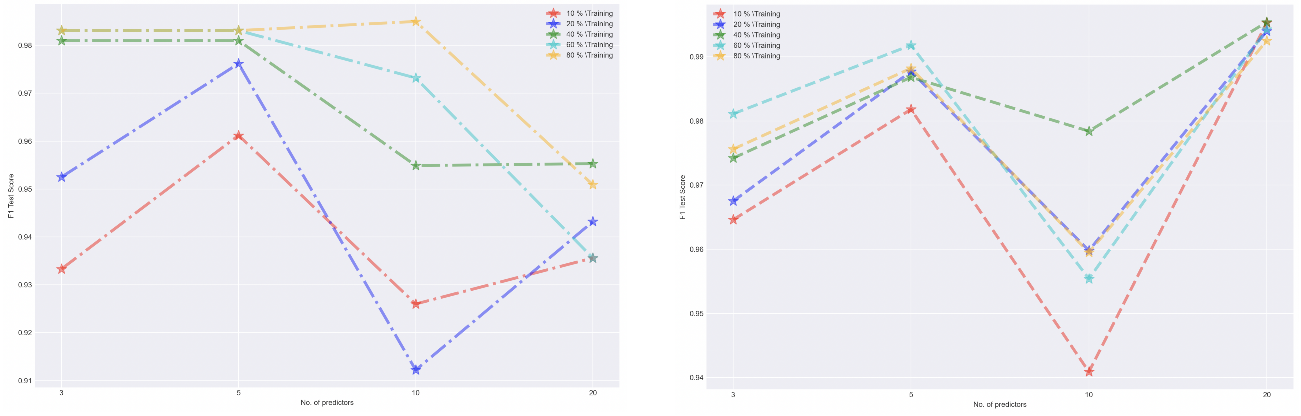
Usually, increasing the sample size improves power. Furthermore, our feature engineered dimension denoted by $N\sigma$ i.e., *(scaled weight)* also affects the power significantly. We can observe the correlation values concerning power in Fig. 13. Thus, we expect the variance covered in PCA transformed data to require at least two dimensions.



(a)

(b)

Figure 10: **PNN REG Classification Performance:** We demonstrate how the change in model complexity due to increase in predictors affects PNN's performance. ***(left)*** (classification boundary 0.8) and ***(right)*** (classification boundary 0.6).



(a)

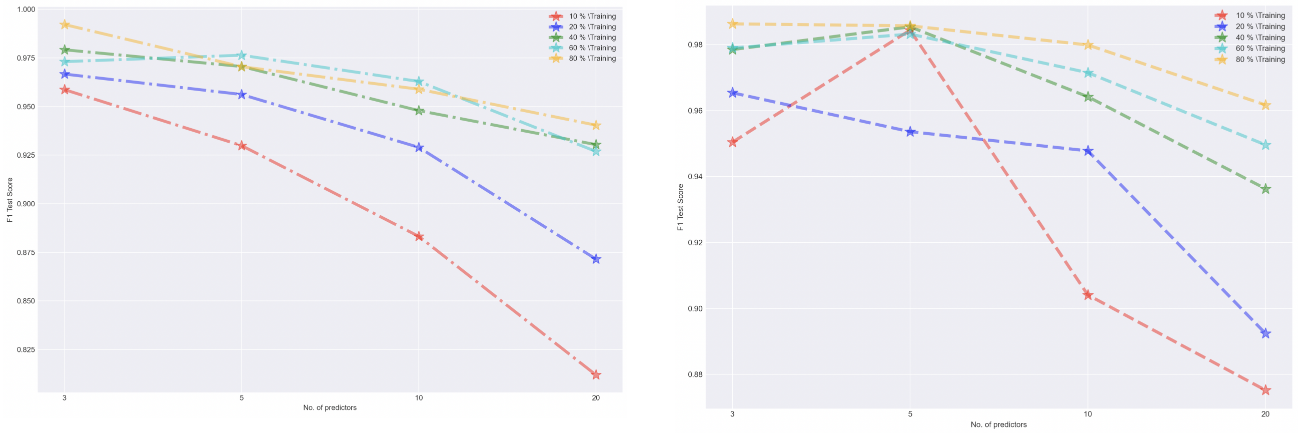
(b)

Figure 11: **PNN RMANOVA Classification Performance:** We demonstrate how the change in model complexity due to increase in predictors affects PNN's performance. ***(left)*** (classification boundary 0.8) and ***(right)*** (classification boundary 0.6).

Experimentation Details	Values
No. of power simulations for each record	1000
Total Datapoints per model	2000
No. of training epochs	500
Training Data Split	10, 20, 40, 60, 80 (%)
Power Classification Boundary	0.6, 0.8
Optimizer	Adam
Model Performance confidence interval	95%

Table 3: Experimental Parameters

We consistently observe excellent performance after selecting 99% variance from PCA. Furthermore, we concatenate these new features to the original matrix and use this to train the Neural Network. As we see from figure 13, the impact of the



(a)

(b)

Figure 12: **PNN LOGIT Classification Performance:** We demonstrate how the change in model complexity due to increase in predictors affects PNN's performance. ***(left)*** (classification boundary 0.8) and ***(right)*** (classification boundary 0.6).

PCA features is sizeable.

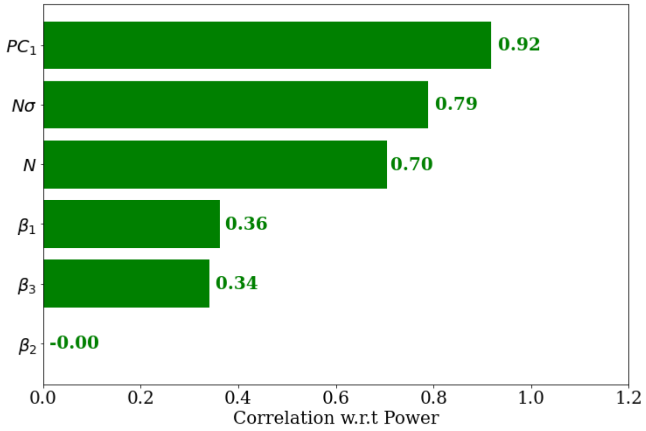


Figure 13: **Impact of PCA:** Correlation of parameters with *true* power. With feature engineering and PCA we are able to obtain correlation > 0.9 for a linear regression model with 3 parameters.

B.1.3. Choice of clusters in Power Cluster

Depending on how the data is clustered, the Choice of several clusters can directly impact the classification performance. However, empirically this method is unreliable and requires a visual inspection of the dataset to be helpful.

B.1.4. Other Considerations

We employ a few techniques during neural network training with increasingly complex models and parameter spaces. (1) The training is sensitive to the choice of the learning rate. Tuning needs to be achieved after using a hyperparameter tuner like Optuna (Akiba et al., 2019). (2) Training tends to fluctuate a lot, and a higher number of epochs and early stopping seem to help.

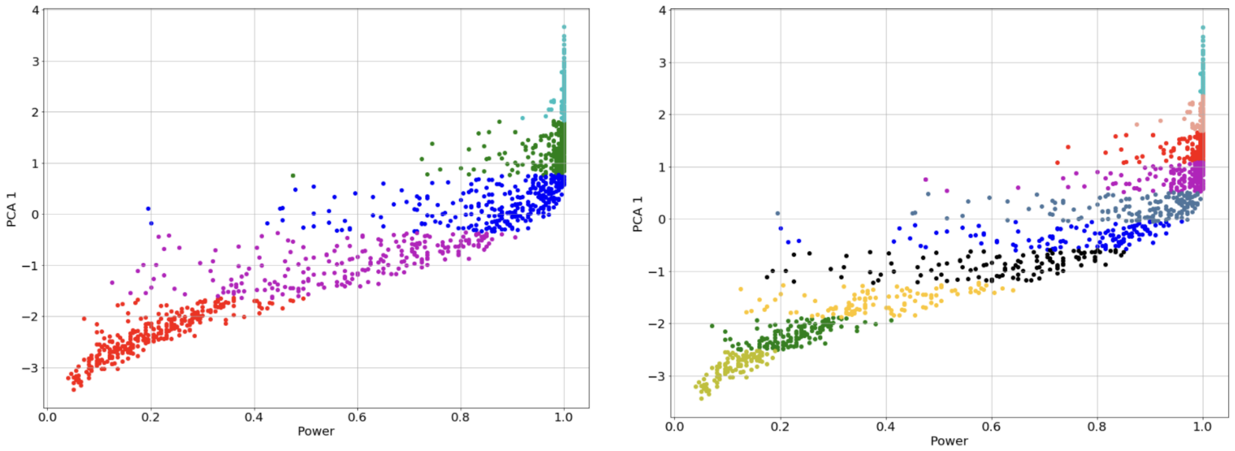


Figure 14: **P-CLUSTER Performance (multi-clustering):** P-CLUSTER can identify multiple zones in the power domain quite efficiently. *(left)* displays 5 clusters and, *(right)* displays 10 clusters.

B.1.5. Neural Network Architecture

Layer	Parameters
Dense (input)	64 units + ReLU
Dense (hidden)	32 units + ReLU
Dense (output)	1 unit + Sigmoid

Table 4: *PNN* Architecture: We report the fully connected layers. The input dimensions depend on the number of predictors and additional PCA features.

B.1.6. List of Model Parameters

D_{X_1}	D_{X_2}	D_{X_3}	D_{X_4}	D_{X_5}
CAT[-1, 1]	$\mathcal{N}(0,1)$	$X_1 \times X_2$	$\mathcal{N}(0,2)$	$X_4 \times X_2$
D_{X_6}	D_{X_7}	D_{X_8}	D_{X_9}	$D_{X_{10}}$
CAT[0, 1, 2]	$\mathcal{N}(0,2)$	$X_6 \times X_7$	$\mathcal{N}(0,1)$	$X_2 \times X_6$
$D_{X_{11}}$	$D_{X_{12}}$	$D_{X_{13}}$	$D_{X_{14}}$	$D_{X_{15}}$
$\mathcal{N}(0,3)$	$\mathcal{N}(0,1)$	$X_1 \times X_{11}$	$\mathcal{N}(0,2)$	$X_{11} \times X_{12}$
$D_{X_{16}}$	$D_{X_{17}}$	$D_{X_{18}}$	$D_{X_{19}}$	$D_{X_{20}}$
$\mathcal{N}(0,2)$	$\mathcal{N}(0,2)$	$X_{11} \times X_{14}$	$\mathcal{N}(0,1)$	$X_6 \times X_{16}$

Table 5: $D_{\mathcal{O}}$: Distribution of model features. For predictors $k < 20$, we only select the first k predictors. Note that CAT refers to a categorical variable while $\mathcal{N}(\mu, \sigma)$ is the Normal Distribution with mean μ and standard deviation σ.

D_{X_1}	D_{X_2}	D_{X_3}	D_{X_4}	D_{X_5}
$\mathcal{N}(0,1)$	CAT[-1, 1]	$\mathcal{N}(0,1)$	$\mathcal{N}(0,1)$	$X_2 \times X_1$
D_{X_6}	D_{X_7}	D_{X_8}	D_{X_9}	$D_{X_{10}}$
$X_2 \times X_3$	$X_2 \times X_4$	$\mathcal{N}(0,1)$	$\mathcal{N}(0,1)$	$X_2 \times X_8$
$D_{X_{11}}$	$D_{X_{12}}$	$D_{X_{13}}$	$D_{X_{14}}$	$D_{X_{15}}$
$\mathcal{N}(0,3)$	$\mathcal{N}(0,1)$	$X_1 \times X_{11}$	$\mathcal{N}(0,2)$	$X_{11} \times X_{12}$
$D_{X_{16}}$	$D_{X_{17}}$	$D_{X_{18}}$	$D_{X_{19}}$	$D_{X_{20}}$
$\mathcal{N}(0,2)$	$\mathcal{N}(0,2)$	$X_{11} \times X_{14}$	$\mathcal{N}(0,1)$	$X_6 \times X_{16}$

Table 6: $D_{\mathcal{A}}$: Alternate Distribution of model features (used for non-linear models and testing purposes). For predictors $k < 20$, we only select the first k predictors. Note that CAT refers to a categorical variable while $\mathcal{N}(\mu, \sigma)$ is the Normal Distribution with mean μ and standard deviation σ.

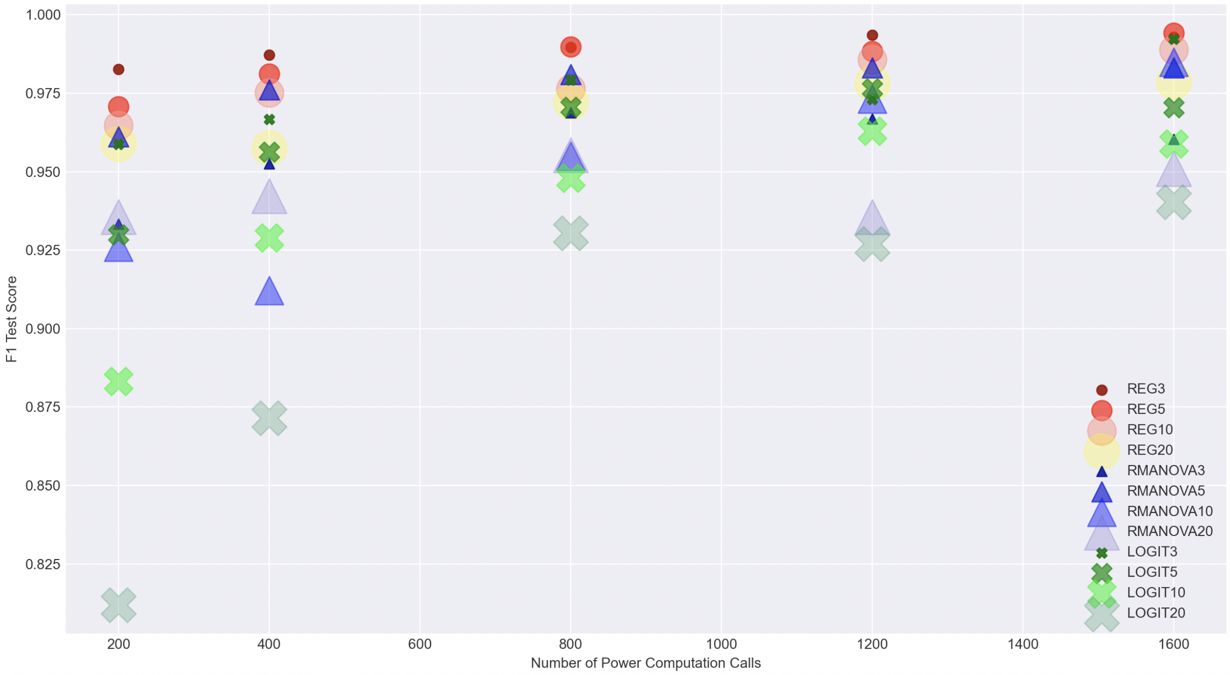


Figure 15: Cost of Power Computation: We report the calls required to compute the power manifold over a subset of the parameter space. Visualizing from left to right, we can see that we can yield high-performing classifiers even with a fraction of the training data.

C. Computational Costs

C.1. Details about Principal Component Analysis

PCA works by finding the direction with maximal variance. Next, it finds the direction with the maximal variance such that this direction is uncorrelated with the previous direction. We continue in this fashion so that any pair of directions are uncorrelated. We then reorient our dataset with these new components to compute our new dataset. Suppose our original dataset is $X \in \mathbb{R}^{N \times p}$ and $v \in \mathbb{R}^{p \times k}$ are the derived Principal Components (PCs) where N is the total samples in X,

p =dim(X) and k is the number of PCs to be selected. Then the new dataset is given by,

$$X^{'} = X \times v.$$