Predicting Phase 3 Clinical Trial Results by Modeling Phase 2 Clinical Trial Subject Level Data Using Deep Learning

Youran Qi YQI28@WISC.EDU
Department of Statistics
University of Wisconsin-Madison
Madison, WI, USA

Qi Tang QI.TANG@SANOFI.COM
Digital and Data Sciences
Sanofi
Bridgewater, NJ, USA

Abstract

Predicting Phase 3 clinical trial results is a critical step of Go/No-Go decision making and Phase 3 trial design optimization. To predict the overall treatment effect for patients enrolled into a Phase 3 trial, we propose a framework consisting of two models. First, an individual trough pharmacokinetic concentration (Ctrough) model is developed to predict the trough pharmacokinetic concentration for a potentially new treatment regime planned for Phase 3. Second, an individual treatment effect model is built to model the relationship between patient baseline characteristics, Ctrough and clinical outcomes. These two models are combined together to predict Phase 3 clinical trial results. Since the clinical outcomes to be predicted are longitudinal and the predictors are a mix of time-invariant and time-variant variables, a novel neural network, Residual Semi-Recurrent Neural Network, is developed for both models. The proposed framework is applied in a post-hoc prediction of Phase 3 clinical trial results, and it outperforms the traditional method.

1. Introduction

Recent failures of drugs (Harrison, 2016) with novel mechanisms of action in Phase 3 clinical trials suggest that our understanding of human biology is still limited and that the traditional model-informed drug development (MIDD) approach needs to be improved. US Food and Drug Administration (FDA) recently published 22 case studies where Phase 3 and Phase 2 had divergent results (FDA, 2017). Although some of the divergent examples were due to failure of translation of drug benefit seen on short term endpoint to long term endpoint, several of them had consistent endpoints but still failed in Phase 3. This is due to the biased prediction of Phase 3 results caused by multiple confounding effects in Phase 2, inadequate model assumptions, and/or the population shift between Phase 2 and Phase 3. It indicates that traditional MIDD methods which focused on population level prediction with manual feature extraction may not be able to predict Phase 3 results reliably based on early phase clinical trial data with a small sample size.

To solve this problem, we propose to predict Phase 3 trial results based on the aggregation of predicted individual treatment effects with an automated feature extraction

procedure. It consists of two deep learning models. First, given the baseline characteristics of Phase 3 patients, their trough concentrations (Ctrough) can be predicted by a Ctrough model, which models the pharmacokinetic (PK) relationship between the dose level and the exposure of the active ingredient of the treatment. This is a critical step because the PK exposure level is a key factor that drives a patient's clinical response, and such a model can be used to predict PK concentrations for the Phase 3 patients even when the Phase 3 treatment regimens are different from those in Phase 2 (Mould and Upton, 2012). With both the baseline characteristics and predicted Ctrough, an individual treatment effect (ITE) model can be built using the Phase 2 subject level data to predict ITEs for the Phase 3 patients. Finally, the ITEs will be aggregated to provide a prediction of the average treatment effect (ATE) for the Phase 3 trial. The whole procedure can be replicated multiple times to obtain a prediction confidence interval and probabilities of success for all outcomes of interest.

We propose to use deep learning models due to the following three reasons.

1. Although traditional population PK (PopPK) models can provide accurate estimate to the average trough concentration for the whole population (FDA, 2005), deep learning models (also called neural networks or artificial neural networks) (LeCun et al., 2015) are shown to be able to provide more accurate predictions for individual patient PK concentration (Brier et al., 1995). The PK relationship can be heterogeneous, highly nonlinear, and impacted by many patient characteristics, so it is often challenging for traditional models to accurately capture the complex nonlinear relationship at subject level, and we often have to construct the desired features by ourselves with expert knowledge, which is a nontrivial task (Mould and Upton, 2012). However, according to the universal approximation theorem (Hornik et al., 1989), a feed-forward neural network with a single hidden layer can well capture complex nonlinear relationships, defined by any continuous functions on compact subsets of an Euclidean space. Actually, neural networks can automatically construct the desired features without any expert knowledge (Bengio et al., 2013).

2. Deep learning methods enable us to use an unified framework to predict the Ctrough and individual treatment effects. Traditionally, NONMEM (Beal et al., 1992) is used to fit a nonlinear mixed effects model based on differential equations (Sheiner and Beal, 1980) for pharmacokinetic/pharmacodynamics (PK/PD) analysis, and then the fitted model is transferred manually from NONMEM to another computing platform, say R (R Core Team, 2017) or Python, for integration with Phase 3 clinical trial design to predict clinical trial results. The proposed deep learning framework connects these two tasks together and completes them using the same kind of model within one computing platform.

3. Although the size of the clinical trial data is relatively small, the signal-to-noise ratio is relatively large, which makes it possible to train a deep learning model with a relatively small data set.

In addition, we decide to use the recurrent neural network (RNN), instead of the feed-forward network, because the clinical trial data are longitudinal data and it is critical to take account of the correlation between time steps, which can be achieved by RNN (Choi et al., 2015, 2016). Furthermore, the size of the clinical trial data is usually small, but the

predictors in the data are often a mix of many time-invariant variables (e.g., patient baseline characteristics) and few time-variant variables (e.g., dose level and/or Ctrough). To efficiently analyze this special kind of data, we tailor the traditional RNN model and propose a novel RNN - Residual Semi-Recurrent Neural Network (RS-RNN). Roughly speaking, RS-RNN feeds the large number of time-invariant predictors to the initial time step of the recurrent layer only, instead of feeding them to every time step of the recurrent layer. Beyond that, RS-RNN further creates a shortcut to directly propagate the information contained in the time-invariant predictors to the final output and facilitate the backward propagation of errors. We will detailedly introduce the proposed RS-RNN in Section 3.2.

Technical Significance We pioneer the use of RNN to model pharmacokinetic concentrations. The RNN model may yield better predictions of individual patient PK concentrations, and thus a better prediction of overall clinical trial result by aggregating individual predictions. To handle the special structure of the small-scale clinical trial data, we propose a novel deep learning model — Residual Semi-Recurrent Neural Network by tailoring the traditional RNN model. Compared with the traditional RNN, RS-RNN reduces model complexity, utilizes information contained in the longitudinal data more efficiently, and models the relationship between responses and predictors (consisting of time-invariant and time-variant variables) in a more intuitive manner.

Clinical Relevance The proposed framework can provide more accurate predictions of clinical trial results, which assists clinical project teams to make informed Go/No-Go decisions for reducing drug failure rate in Phase 3 and to better design Phase 3 trials, since the proposed framework carefully considers the key factors in the design of Phase 3 trial, e.g., dose level and dosing frequency.

2. Cohort

For the purpose of not disclosing confidential information about the investigational product, limited details are provided here.

2.1. Cohort Selection

Patients were enrolled into a double-blind randomized Phase 2 clinical trial for evaluation of the efficacy and safety of an investigational product. Similar inclusion and exclusion criteria were used for a confirmatory Phase 3 clinical trial, of which the clinical trial results were blinded to the modeler who applied the proposed framework to the Phase 3 trial, to avoid cherry picking in the model optimization procedure.

2.2. Data Extraction

The outcomes of interest are patients' responses to an investigational treatment measured by change from baseline of a physical function at Week 16. Patients' baseline characteristics pre-specified in the primary and exploratory analyses of the Statistical Analysis Plan of the Phase 2 study were included as time-invariant predictors for the response variable. Dose level and Ctrough of each patient in the Phase 2 trial were time-variant predictors since

they change week by week. The above subject level data of all patients in Phase 2 were used to build the models.

2.3. Feature Choices

Patient disease history, medication history, demographic information, clinical trial participation date, lab test results and disease related biomarkers measured at baseline before randomization were included as the features for the prediction task. The choice of each feature follows the principle that any variable included in the exploratory and primary analyses of the Statistical Analysis Plan for the Phase 2 study was included.

3. Methods

3.1. The Proposed Framework

As mentioned above, our framework consists of two models. The first model is the Ctrough model, which predicts the Ctrough for Phase 3 patients and can be trained using Phase 2 clinical trial data. The second model is the ITE model, which predicts ITE with Ctrough and baseline profiles as input and can be trained using observed subject level data from the Phase 2 trial. Deep learning methods are used to build both the two models.

First, we build the trough concentration model to predict the Ctrough for given baseline characteristics and dose levels. More specifically, we train and validate the trough concentration model $\hat{\phi}$ using $\{c_i^{(2)}, \tilde{x}_i^{(2)}\}_{i=1}^k$, where $\tilde{x}_i^{(2)} = (x_i^{(2)}, d_i^{(2)})$, $x_i^{(2)}$ is a p-dimensional vector representing the p baseline characteristics of the ith patient, $d_i^{(2)}$ is the sequence of actual dose levels assigned to the ith patient, and $c_i^{(2)}$ is the sequence of observed Ctrough of the ith patient. We want to emphasize that $\hat{\phi}$ is trained and validated using only the patients in the finished Phase 2 clinical trial who will be assigned to the training set used to train the following ITE model and have at least one observed nonzero Ctrough (k in the above expression is just the number of such patients).

Next, we predict the ITE for patients in the Phase 3 clinical trial and the average treatment effect with the following four steps:

1. Train, validate and test the ITE model $\hat{f}$ using $\{y_i^{(2)}, \bar{x}_i^{(2)}\}_{i=1}^n$, where n is the number of patients in the finished Phase 2 clinical trial, $\bar{x}_i^{(2)} = (x_i^{(2)}, d_i^{(2)}, c_i^{(2)})$, the missing values in $c_i^{(2)}$ are imputed by $\hat{\phi}((x_i^{(2)}, d_i^{(2)}))$, and $y_i^{(2)}$ is an r-dimensional response vector consisting of the r endpoints of the ith patient in the finished Phase 2 clinical trial (in this paper, we focus on only one endpoint so $r = 1$).

2. Apply the ITE model $\hat{f}$ to $\bar{x}_j^{(3)}$ to get $\hat{y}_j^{(3)} = \hat{f}(\bar{x}_j^{(3)})$ for $j = 1, \ldots, m$, where $\bar{x}_j^{(3)} = (x_j^{(3)}, d_j^{(3)}, \hat{c}_j^{(3)})$, $x_j^{(3)}$ is a p-dimensional vector representing the p baseline characteristics of the jth patient in the Phase 3 trial, $d_j^{(3)}$ is the dose level we plan to assign to the jth patient in the Phase 3 trial, $\hat{c}_j^{(3)} = \hat{\phi}((x_j^{(3)}, d_j^{(3)}))$, and $\hat{y}_j^{(3)}$ is the predicted response of the jth patient when the patient receives dose level $d_j^{(3)}$.

3. Let the $d_j^{(3)}$ and $\hat{c}_j^{(3)}$ in Step 2 be 0, and then repeat Step 2 to get $\tilde{y}_j^{(3)}$, which is the predicted response of the jth patient when the patient receives placebo.

4. The ITE of the jth patient in the Phase 3 clinical trial is $\hat{y}_j^{(3)} - \tilde{y}_j^{(3)}$, and the average treatment effect of the Phase 3 clinical trial is $\sum_{j=1}^{m}(\hat{y}_j^{(3)} - \tilde{y}_j^{(3)})/m$.

Step 1 involves training the model on the training set, fine-tuning the hyperparameters of the model on the validation set and testing the prediction performance on the test set. Steps 1 and 2 are illustrated by Figure 1. In addition, we require that $x_i^{(2)}$ consists of the same baseline characteristics as $x_j^{(3)}$.

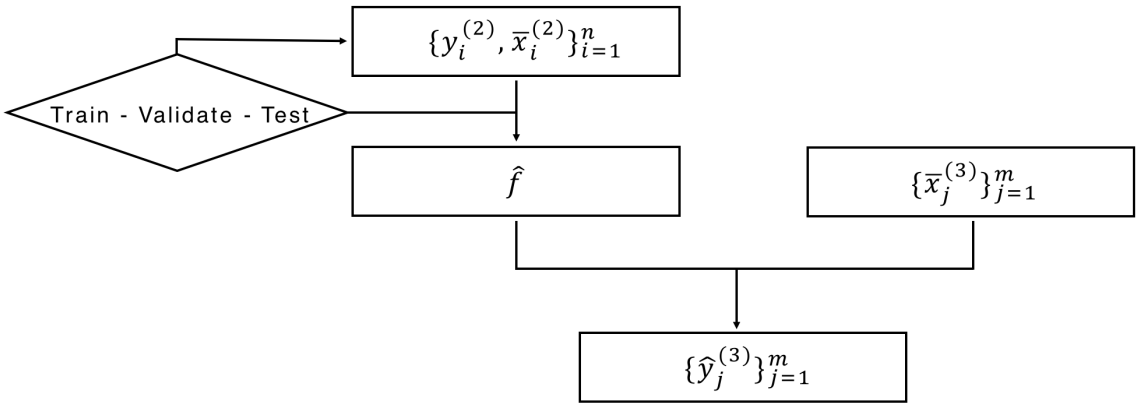

Figure 1: Steps 1 and 2 for ITE prediction.

3.2. Residual Semi-Recurrent Neural Network (RS-RNN)

Deep learning is a machine learning method where we use a multiple-layer neural network to fit the relationship between the response and predictors. In this section, we will introduce the traditional RNN and the novel RNN we proposed to build the above-mentioned Ctrough model and ITE model. We also introduce the hyperparameter optimization techniques we used to fine-tune RS-RNN in the Appendix.

3.2.1. Recurrent Neural Network (RNN)

The recurrent neural network is a class of functions f defined as follows. For a given instance i and a sequence of inputs $\{x_{it}\}_{t=1}^{T_i}$, the sequence of outputs $\hat{y}_{it} = f(x_{it})$ can be obtained by

$$\begin{aligned} s_{i0} = 0, \quad & s_{it} = \sigma(Ux_{it} + Ws_{i,t-1} + b), \quad && t = 1, \ldots, T_i \\ & \hat{y}_{it} = \sigma(Vs_{it} + a), && t = 1, \ldots, T_i, \end{aligned}$$

where the first equation defines a recurrent layer, the second equation defines a fully-connected layer, σ is a component-wise activation function, $x_{it} \in \mathbb{R}^p$, $s_{it} \in \mathbb{R}^q$, $\hat{y}_{it} \in \mathbb{R}^r$ and $U \in \mathbb{R}^{q\times p}$, $W \in \mathbb{R}^{q\times q}$, $b \in \mathbb{R}^q$, $V \in \mathbb{R}^{r\times q}$, $a \in \mathbb{R}^r$ are the parameters we need to train. Training an RNN means finding a set of parameters U, W, b, V, a, or equivalently a

particular function f, to make $\hat{y}_{it}$ close to the observed response y_{it}. This means we want to solve the optimization problem

$$\min_{U,W,b,V,a} \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{T_i} L(\hat{y}_{it}, y_{it}),$$

where n is the number of training instances, and L is a pre-specified loss function. This optimization problem is often solved by the stochastic gradient descent (SGD).

We want to emphasize that the above definition is more like a general framework and it actually has tons of variants. For example, the output does not have to be a sequence over time and one can use the output at a particular time step only, e.g., the last time step $\hat{y}_{iT_i} \in \mathbb{R}^r$. There could also be multiple layers, i.e., one can regard s_{it} as x_{it} and then stack one more recurrent layer on top of s_{it}, or one can also regard $\hat{y}_{it}$ as s_{it} and then stack one more fully-connected layer on top of $\hat{y}_{it}$. There are also different ways to define the recurrent layer, e.g., regular recurrent layer, LSTM layer, GRU layer, etc. Furthermore, one may also use different activation functions, use different numbers of neurons in each layer, and so on. All of these are called hyperparameters of a recurrent neural network. Such a flexibility enables the recurrent neural network to handle various kinds of problems but it also incurs a problem of choosing those hyperparameters - a bad hyperparameter combination will lead to very poor prediction result. We introduce how to find a good hyperparameter combination in the Appendix.

3.2.2. Residual Semi-Recurrent Neural Network (RS-RNN)

In the framework introduced in Section 3.1, suppose we directly apply the above-mentioned traditional RNN model. When we train $\hat{\phi}$ and $\hat{f}$, for a given instance i, the sequence of inputs is $\{(x_{it}^{(2)}, d_{it}^{(2)})\}_{t=1}^{T_i}$ and $\{(x_{it}^{(2)}, d_{it}^{(2)}, c_{it}^{(2)})\}_{t=1}^{T_i}$ respectively, where $x_{i1}^{(2)} = x_{i2}^{(2)} = x_{i3}^{(2)}, \ldots, = x_{iT_i}^{(2)}$ actually do not change over time since they are baseline characteristics. In other words, we actually only have two time-variant predictors - the does level and Ctrough, but in order to use the traditional RNN model, we have to regard the time-invariant baseline characteristics also as time-variant predictors, which creates a large amount of duplicated data, unnecessarily makes the RNN model too complicated, and leads to poor prediction performance. This will be a more serious problem in the high-dimensional scenarios where we have a large number of time-invariant predictors, which is exactly the case in our application (after representing the categorical predictors using one-hot vectors, we have 50 time-invariant predictors).

In order to handle this special kind of data - the longitudinal data with a large number of time-invariant predictors and few time-variant predictors, we propose the RS-RNN. The key idea of RS-RNN is to divide the high-dimensional longitudinal data into two parts: one part consists of all the time-invariant predictors and the other part consists of all the time-variant predictors. Next, we feed the time-invariant predictors to a multilayer perceptron (MLP), which is nothing but multiple stacked fully-connected layers. Then we use the output of this MLP as the initial state of the traditional RNN layer, and feed the time-variant predictors into the traditional RNN layer as usual. In addition, it is also possible that the time-invariant predictors are more influential than the time-variant predictors. In order to take care of this case, we create shortcut connections between the MLP used to

handle the time-invariant predictors and the fully-connected layers stacked on top of the recurrent layers. This will create a shortcut to directly propagate the information contained in the time-invariant predictors to the final output and facilitate the backward propagation of errors, which is very similar to the idea of Residual Network (He et al., 2016). Hence, we call our final model the Residual Semi-Recurrent Neural Network.

Mathematically, the RS-RNN is defined as follows. For a given instance i, the time-invariant predictors x_i and a sequence of time-variant predictors $(z_{i1}, z_{i2}, \ldots, z_{iT_i})$, the sequence of outputs $\hat{y}_{it} = f^{RS}(x_i, z_{i1}, z_{i2}, \ldots, z_{iT_i})$ is obtained by

$$\begin{aligned} s_{i0} &= \sigma(Tx_i + c), \quad & s_{it} &= \sigma(Uz_{it} + Ws_{i,t-1} + b), \quad & t &= 1, \ldots, T_i, \\ & & \hat{y}_{it} &= \sigma(Vs_{it} + a) + Ps_{i0}, & t &= 1, \ldots, T_i, \end{aligned}$$

where $s_{i0} = \sigma(Tx_i + c)$ is a fully-connected layer of the MLP to handle the time-invariant predictors, $x_i \in \mathbb{R}^k$, $z_{it} \in \mathbb{R}^p$, $s_{it} \in \mathbb{R}^q$, $\hat{y}_{it} \in \mathbb{R}^r$ and $T \in \mathbb{R}^{q \times k}$, $c \in \mathbb{R}^q$, $U \in \mathbb{R}^{q \times p}$, $W \in \mathbb{R}^{q \times q}$, $b \in \mathbb{R}^q$, $V \in \mathbb{R}^{r \times q}$, $a \in \mathbb{R}^r$, $P \in \mathbb{R}^{r \times q}$ are the parameters we need to train. For the Ctrough model, we have $x_i = x_i^{(2)}$ and $z_{it} = d_{it}^{(2)}$, while for the ITE model, we have $x_i = x_i^{(2)}$ and $z_{it} = (d_{it}^{(2)}, c_{it}^{(2)})$. We illustrate the RS-RNN in Figure 2.

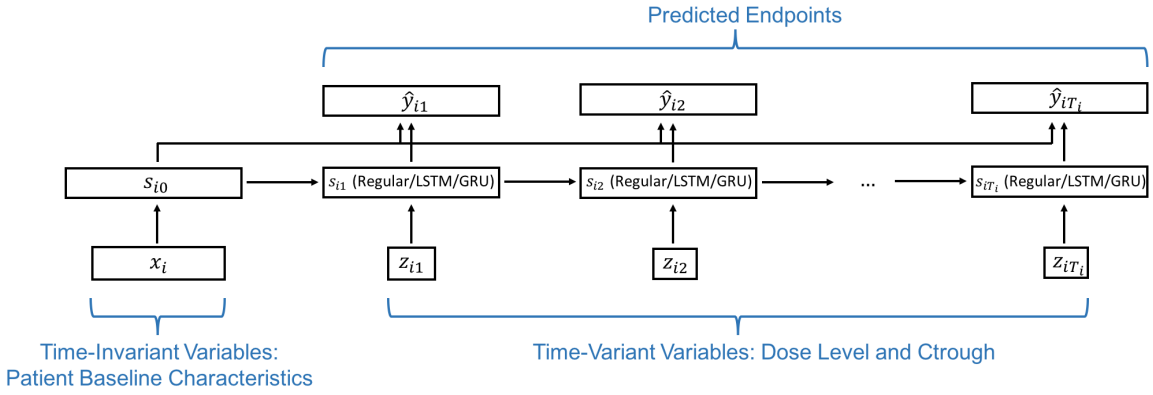

Figure 2: The proposed RS-RNN.

The MLP to handle the time-invariant predictors may contain more than one fully-connected layers. In addition, when there are more than one recurrent layers, we may feed the output of the MLP to the first recurrent layer only, or the last recurrent layer only, or all the recurrent layers. Similarly, when there are more than one fully-connected layers on top of the recurrent layers, we may add the output of the MLP to the first fully-connected layer only, or the last fully-connected layer only, or all the fully-connected layers. We regard these different choices as hyperparameters.

4. Results

4.1. Evaluation Approach/Study Design

We will demonstrate the superiority of RNN and the proposed RS-RNN by two real applications. In the first application, we apply RNN to the PK modeling problem. In our data,

each patient has a long sequence of observed PK values (> 30 time steps). We use the root of mean squared prediction error (RMSPE) as the evaluation criterion to measure how well we fit the PK curves.

In the second application, we apply the proposed RS-RNN to predict the clinical trial results for a drug. As mentioned in Section 2.2, we focus on only one endpoint — the change from baseline of a physical function at Week 16 and use it as our response. The treatment of interest is the low dose regime, which is coded as "low dose" arm. We train the proposed RS-RNN by the Phase 2 clinical trial data consisting of 20 time-invariant predictors (they become 50 predictors after representing the categorical ones using one-hot vectors) and 2 time-variant predictors - dose level and Ctrough. We test the model using the Phase 3 clinical trial data consisting of a large number of patients and the same set of predictors.

4.2. Application 1: PK Modeling

Pharmacokinetic modeling is a critical step of drug development, which models how a drug interacts with human body, i.e., the time course of drug absorption, distribution, metabolism, and excretion. Traditionally, differential equations borrowed from the field of fluid dynamics were used to model such interactions between drug and human body. For example, multiple-compartment differential equations models (Mortensen et al., 2008) were used to model the PK curves on the data set we are considering. Differential equations can describe the underlying physical process inside human body after being dosed. However, they may not be able to take into account all the heterogeneities between patients. Therefore, the PK prediction for each individual patient given by this method may be poor, though the prediction at population level could be accurate.

For each patient in our data set, there is a sequence of observed PK values, two baseline characteristics - age and sex, and a sequence of cumulative amounts of drug injected. They correspond to the $c_i^{(2)}$, $x_i^{(2)}$ and $d_i^{(2)}$ in Section 3.1. The fitting result of RNN on a validation set with 12 patients is shown in Figure 3.

Note that the RNN model accurately fits the PK curve for each of the 12 patients, where there are two PK curves with very high peaks that have not been seen in the training set. The RMSPE on the validation set is 12, which is very small when considering the range of PK values. This indicates that the RNN model is able to capture the heterogeneous and complex nonlinear relationship between the PK values, and the cumulative amounts of drug injected and the baseline characteristics. This example shows that it is promising to perform the PK modeling using deep learning models like RNN.

4.3. Application 2: Clinical Trial Prediction

According to the results of a Phase 2 clinical trial, the drug can significantly improve patients' health. However, after a Phase 3 trial, people found its effect shrunk. This may be due to many reasons, such as population shift, change of endpoints, change of treatment regime, etc.. With the help of the proposed framework, the reason of inconsistent treatment effects can be better examined, because the framework adopts the deep learning methods that can capture very complex relationships.

As mentioned in Section 3.1, we will build a Ctrough model and an ITE model, and feed the observed Phase 3 patients' baseline characteristics and treatment regimes into the

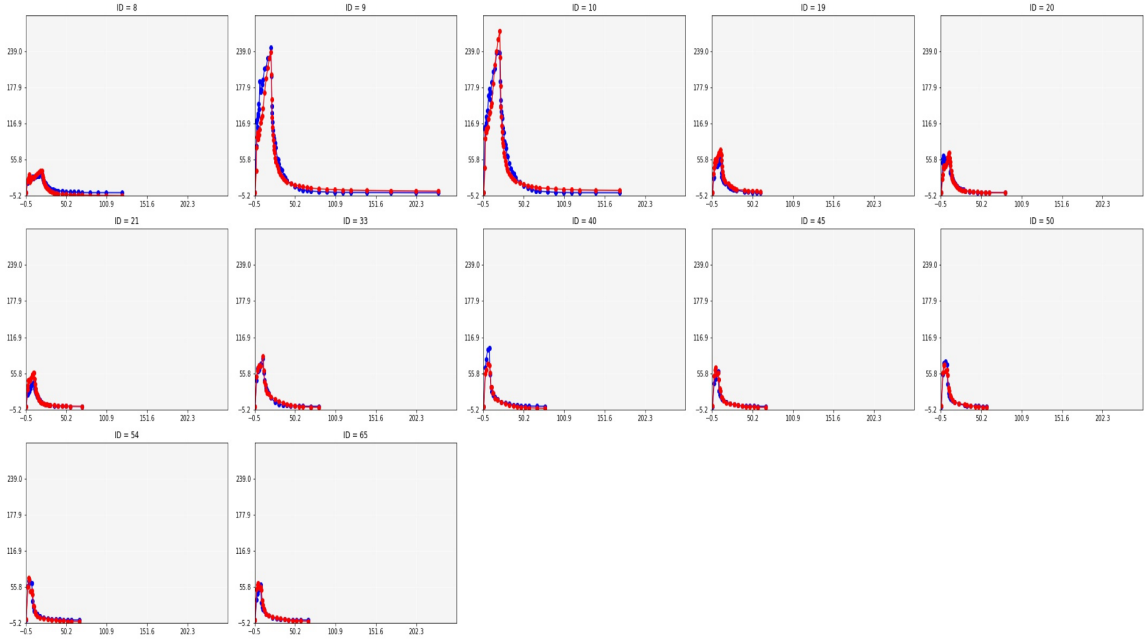

Figure 3: Fitting result of RNN on a validation set with 12 patients (blue curves are observed PK curves and red curves are predicted PK curves).

Ctrough model and the ITE model to predict the Phase 3 trial results. The predicted average treatment effects (ATE) are displayed in Figure 4. Specific hyperparameters of the Ctrough model and ITE model can be found in Table 1 in the Appendix.

Looking at the two bars on the right of Figure 4, we find that without using the observed Phase 3 responses (i.e., only using Phase 2 data and Phase 3 data excluding responses), RS-RNN predicts that the ATE for the "low dose" arm in the Phase 3 clinical trial is 1.58. This is very close to the ATE predicted by a mixed effect model using the observed Phase 3 responses 1.61. We believe the value 1.61 given by the mixed effect model should be very close to the true ATE of the drug, because it is obtained by using the large number of observed Phase 3 responses. This indicates that the proposed framework with RS-RNN can accurately predict the ATE in the Phase 3 clinical trial after all patients finish enrollment but before unblinded results are obtained, which is useful for blinded clinical trial monitoring and decision making.

Looking at the two bars on the left of Figure 4, we find that when using the Phase 2 data only, RS-RNN gives a prediction of 1.63, which is very close to the true ATE of Phase 3 trial and its own prediction without using the observed Phase 3 responses 1.58. This makes sense since the distributions of the patients in the two trials are similar. However, when using the Phase 2 data only, the traditional mixed effect model predicts that the ATE is 2.27, which is much higher than the true ATE. This indicates that the traditional method used to analyze the Phase 2 data fails to identify important confounding factors and model their relationships.

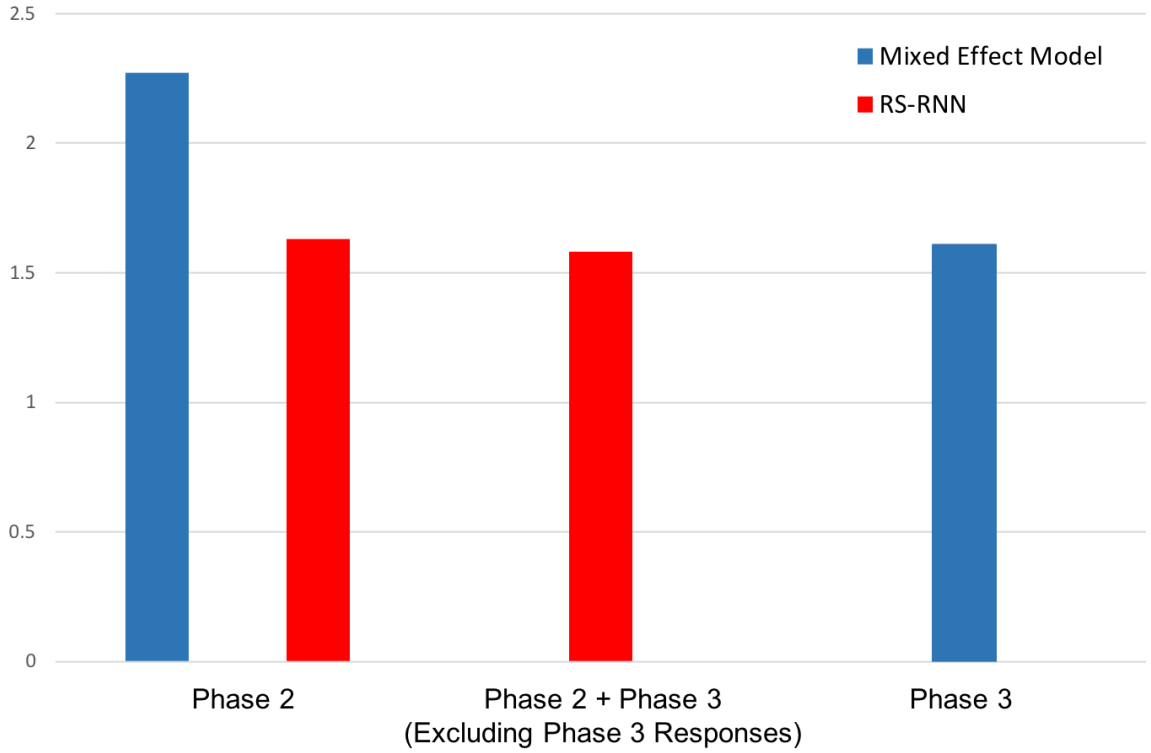

Figure 4: The predicted average treatment effect (ATE).

Finally, we interpret the prediction results of RS-RNN for the Phase 3 trial by the GUIDE regression tree (Loh, 2002) in Figure 5.

5. Discussion and Related Work

Translation from Phase 2 trial to Phase 3 trial is challenging. In this paper, we propose a new framework to predict the Phase 3 clinical trial results based on the Phase 2 clinical trial data. Under this framework, we further propose a novel RNN model, RS-RNN, to predict the Ctrough and treatment effects in the Phase 3 clinical trial. We showed by a real application that the new framework and the proposed deep learning method significantly outperform the traditional method in terms of predicting the Phase 3 results. In addition, we pioneer the use of RNN to model pharmacokinetic concentrations. We find the RNN model performs well and is a promising alternative method for PK modeling.

The future work is as follows. First, we will use the bootstrap technique to get a confidence interval for the predicted average treatment effect in the Phase 3 trial. A narrow confidence interval that covers the true average treatment effect will demonstrate the reliability of the proposed method.

Second, now we only use the change from baseline of a physical function at Week 16 as the response and only predict responses in one treatment group in the Phase 3 trial, but we can also use multiple endpoints as the responses and extend the predictions to other

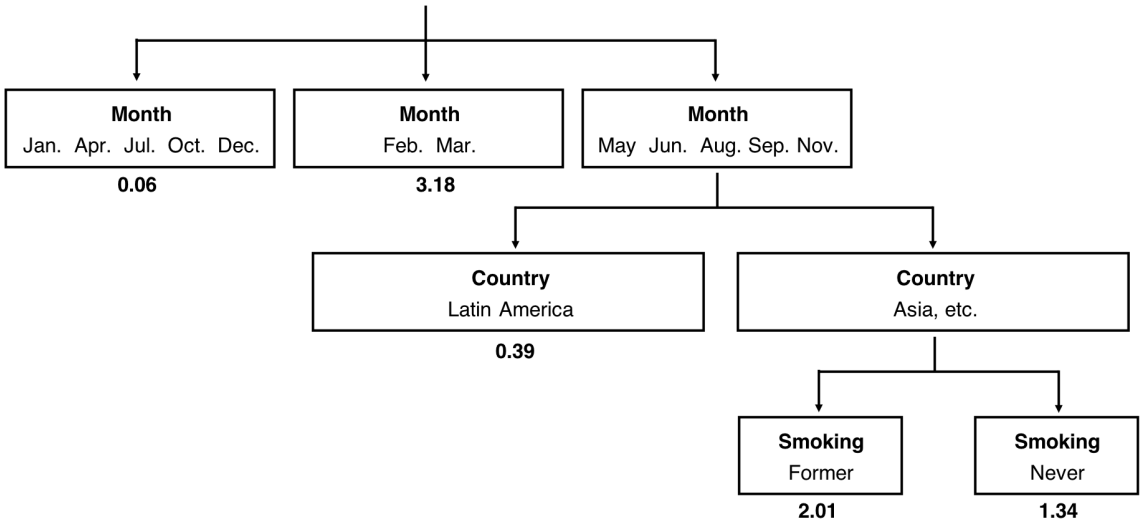

Figure 5: Interpretation by the regression tree: the value below each node is the median of the predicted responses for the patients in the node.

treatment groups. Thanks to the flexibility of the neural network, this is straightforward and incurs little additional work.

Third, the proposed framework needs baseline characteristics of the patients enrolled in Phase 3 trial as the input to the models. If we do not know which patients will enroll in the Phase 3 trial, then we can simulate these patients in the following ways. First, we can randomly resample the Phase 2 patients to obtain the Phase 3 patients (Marshall et al., 2016). However, this method relies on a homogeneity assumption that Phase 3 patients are similar to Phase 2 patients in terms of distributions of baseline characteristics, which is often violated in practice (Hanin, 2017). Actually, as pointed out in the aforementioned FDA report, the Phase 3 population is often much more heterogeneous than Phase 2 population. One possible reason is the change in treatment regime, for example dosing frequency, and another possible reason is the change in the inclusion and exclusion criteria, which brings in new patients with different distributions of characteristics. Second, when the real world information, e.g., electronic health records (EHR), is available, we can obtain the Phase 3 patients by randomly sampling the patients in the EHR data set who meet the Phase 3 inclusion criteria. However, this method assumes the equal probability of enrollment, which may yield a biased estimate to patient baseline profiles in the Phase 3 trial, because some patients, e.g., the patients with more severe conditions and low standard of care, are more likely to participate in a clinical trial than the others.

Therefore, we plan to add to our framework a third deep learning model — the enrollment model, that will predict the probability of enrollment for each patient in the EHR data set who meets the Phase 3 inclusion criteria, and then we sample the patients according to the predicted probabilities. More specifically, we may simulate patients in the Phase 3 clinical trial using the enrollment model (instead of using the true patients as we did) with the following four steps:

1. Get the data set $\{p_i^{(2)}, z_i^{(2)}\}_{i=1}^N$ from the EHR data set, where N is the number of patients who meet the Phase 2 inclusion criteria, $z_i^{(2)}$ is a q-dimensional vector representing the q baseline characteristics of the ith patient who meets the Phase 2 inclusion criteria, $p_i^{(2)}$ is a binary scalar, that can only be 0 or 1, indicating whether the ith patient actually enrolled in the Phase 2 clinical trial (0 if the patient didn't enroll and 1 otherwise).

2. Train, validate and test the enrollment model $\hat{g}$ using $\{p_i^{(2)}, z_i^{(2)}\}_{i=1}^N$.

3. Apply the enrollment model $\hat{g}$ to $z_j^{(3)}$ to get $\hat{p}_j^{(3)} = \hat{g}(z_j^{(3)})$ for $j = 1, \ldots, M$, where M is the number of patients who meet the Phase 3 inclusion criteria, $z_j^{(3)}$ is a q-dimensional vector representing the q baseline characteristics of the jth patient who meets the Phase 3 inclusion criteria, $\hat{p}_j^{(3)}$ is the predicted enrollment probability of the jth patient who meets the Phase 3 inclusion criteria.

4. Sample $\{z_j^{(3)}\}_{j=1}^M$ according to $\hat{p}_j^{(3)}$ (i.e., the jth patient will be selected into the sample with probability $\hat{p}_j^{(3)}$ to get $\{x_j^{(3)}\}_{j=1}^m$, where m is the number of patients in the simulated Phase 3 clinical trial, $x_j^{(3)}$ is a p-dimensional vector consisting of p components of $z_j^{(3)} (p \leq q)$.

Finally, we will also try to incorporate some expert knowledge to further improve the performance of the Ctrough model and the ITE model.

References

Stuart L Beal, Lewis B Sheiner, Alison Boeckmann, and Robert J Bauer. Nonmem users guides. *NONMEM Project Group, University of California, San Francisco*, 1992.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Michael E Brier, Jacek M Zurada, and George R Aronoff. Neural network predicted peak and trough gentamicin concentrations. *Pharmaceutical research*, 12(3):406–412, 1995.

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. *arXiv preprint arXiv:1511.05942*, 2015.

Edward Choi, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Using recurrent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association*, 24(2):361–370, 2016.

FDA. Guidance for industry: Population pharmacokinetics, 2005.

FDA. 22 case studies where phase 2 and phase 3 trials had divergent results, 2017.

Leonid Hanin. Why statistical inference from clinical trials is likely to generate false and irreproducible results. *BMC medical research methodology*, 17(1):127, 2017.

Richard K Harrison. Phase ii and phase iii failures: 2013–2015. *Nat Rev Drug Discov*, 15 (12):817–8, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Wei-Yin Loh. Regression tress with unbiased variable selection and interaction detection. *Statistica Sinica*, pages 361–386, 2002.

SF Marshall, R Burghaus, V Cosson, SYA Cheung, M Chenel, O DellaPasqua, N Frey, B Hamrén, L Harnisch, F Ivanow, et al. Good practices in model-informed drug discovery and development: practice, application, and documentation. *CPT: pharmacometrics & systems pharmacology*, 5(3):93–122, 2016.

Stig Bousgaard Mortensen, Anna Helga Jónsdóttir, Søren Klim, and Henrik Madsen. Introduction to pk/pd modelling-with focus on pk and stochastic differential equations. 2008.

DR Mould and RN Upton. Basic concepts in population modeling, simulation, and model-based drug development. *CPT: pharmacometrics & systems pharmacology*, 1(9):1–14, 2012.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL https://www.R-project.org/.

Lewis B Sheiner and Stuart L Beal. Evaluation of methods for estimating population pharmacokinetic parameters. i. michaelis-menten model: routine clinical pharmacokinetic data. *Journal of pharmacokinetics and biopharmaceutics*, 8(6):553–571, 1980.

Appendix: Hyperparameter Optimization

As mentioned above, hyperparameter optimization is critical in building a deep learning model since a bad hyperparameter combination will lead to very poor prediction results. It is often impossible to enumerate all the hyperparameter combinations and pick the best

one, since there will be too many combinations (for example, 30 hyperparameters with 2 candidate values for each give us 2^{30} possible combinations) and training the deep learning model once with a given hyperparameter combination may already be very time-consuming. There are three commonly-used hyperparameter optimization techniques.

The first technique is simply random search, i.e., we randomly select a small number of hyperparameter combinations from all the possible combinations, train the model with each of the selected combinations, and pick the one that gives the smallest validation loss.

The second technique is the Latin hypercube search, which is based on the Latin hypercube design. Suppose now we have p factors and each factor has k candidate values. A Latin hypercube design is just a $k \times p$ matrix, where the k candidate values appear once and exactly once in each of the columns. This also applies to continuous factors since we can always discretize them. If the factors do not have the same number of candidate values, then we can simply require the k candidate values appear as uniformly as possible. Since the hypercube design achieves the univariate stratification, i.e., it is uniform on each single dimension, it has better space-filling properties and enables us to explore the space more thoroughly to get the best hyperparameter combination.

The third technique is the Sequential Model-Based Global Optimization (SMBO) (Hutter et al., 2011). It searches the space of hyperparameters as follows:

1. Gather some initial hyperparameter combinations $\{h_i\}_{i=1}^{n_0}$, using random search or Latin hypercube search and train the model with each of them to get the validation losses $\{F(h_i)\}_{i=1}^{n_0}$. Let $H = \{(F(h_i), h_i)\}_{i=1}^{n_0}$.
2. For $t = 1, 2, \ldots, T$
   (a) Fit a model M_t using H.
   (b) Select the next hyperparameter combination h^* by $h^* = \text{argmax}_h S(h, M_t)$.
   (c) Train the deep learning model with h^* and get the validation loss $F(h^*)$.
   (d) Update H with $H \cup (F(h^*), h^*)$.

Here, S is a criterion function and F is a function whose input is a hyperparameter combination and output is the validation loss. Evaluating the function F means training the model once, which is often very time consuming. In our project, we used the SMBO to search the hyperparameters for our proposed RS-RNN. More specifically, we use the Latin hypercube search to get the initial hyperparameter combinations. Then we use the Gaussian process model as M_t and the Expected Improvement (EI) as our criterion function, i.e., we select the next hyperparameter combination h^* by

$$h^* = \underset{h}{\text{argmax}}\ E_{Y(h)\sim N(\mu(h),\sigma^2(h))}[\max(y^* - Y(h), 0)],$$

where the mean $\mu(h)$ and variance $\sigma^2(h)$ of the random variable $Y(h)$ at h are given by the Gaussian process model and y^* is the current minimum validation loss. We list some of the key hyperparameters we searched and used to train the Ctrough model and the ITE model in Table 1.

Table 1: Hyperparameters of Ctrough Model and ITE Model for Phase 3 Trial Prediction.

Layers	**Hyperparameters**	**Ctrough Model**	**ITE Model**
	Loss function	MSE	MSE
	Learning rate	0.03	0.01
	Number of epochs	500	500
	Regularization	l_1 regularization with $\lambda = 0.2$	l_2 regularization with $\lambda = 10^{-4}$
	MLP output	Use it as the initial state of the last recurrent layer and add it to all fully-connected layers on top of the recurrent layers	Use it as the initial state of the first recurrent layer and add it to all fully-connected layers on top of the recurrent layers
Fully-connected layers of the MLP	Number of layers	2	8
	Number of neurons	40, 14	46, 44, 42, 42, 38, 32, 28, 28
	Activation function	ELU	ELU
	Batch normalization	Yes	Yes
Recurrent layers	Number of layers	3	1
	Number of neurons	18, 9, 5	25
	Activation function	ELU	ELU
	Recurrent cell	GRU	GRU
Fully-connected layers on top of the recurrent layers	Number of layers	1	3
	Number of neurons	1	23, 7, 1
	Activation function	ELU	ELU
	Batch normalization	Yes	Yes