Using Random Effects to Account for High-Cardinality Categorical Features and Repeated Measures in Deep Neural Networks

GIORA SIMCHONI, SAHARON ROSSET

DEPARTMENT OF STATISTICS AND OR, TEL AVIV UNIVERSITY

35TH CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS (NEURIPS 2021)

Outline

- Motivation
- Linear Mixed Models
- Our approach: LMMNN
- Results
- Convergence
- Extensions

High cardinality categorical features

- Using EMR of hospital patients to predict hospital readmission, the feature **disease** could take 1 out of thousands of values (Lin et. al., 2019)
- Using the CelebA facial images dataset to develop a computer vision model to localize facial features, several images from the same **person** (a.k.a *repeated measures*), over 10K identities (Liu et. al., 2015)
- Predicting the price of a Airbnb rental, ~40K **hosts** in NYC alone (Kalehbasti et al., 2019)

Current solutions

- One-hot encoding (OHE) → q levels to q additional binary features
- Entity embeddings → reduce dimensionality from q to d with a *learned* $D_{q \times d}$ dictionary
- Ignore!
- Other (Hancock and Khoshgoftaar, 2020):
  - Clustering (expert knowledge, "other" strategy, clustering algos)
  - Feature Hashing
  - Supervised numerical encoding

TL;DR

Table 4: Real data features summary table

Dataset	n	q	p	Categorical	y	Input Type	DNN
UKB PA	96K	350	15	job	PA	Tabular	MLP
Drugs	215K	3.6K	10K	drug	rating	Text	LSTM
CelebA	202K	10K	218x178x3	identity	noseX-Y	Images	CNN
Airbnb	50K	40K	196	host	log(price)	Tabular	MLP

Table 5: Real data 5-CV mean test MSEs and standard errors in parentheses.

Dataset	Ignore	OHE	Embeddings	MeNets	LMMNN
UKB PA	**0.812 (.008)**	0.816 (.009)	0.817 (.010)	**0.811 (.009)**	**0.809 (.008)**
Drugs	2.74 (.032)	2.77 (.005)	2.72 (.051)	2.81 (.031)	**2.66 (.006)**
CelebA noseX	**1.68 (.05)**	–	3.6 (.3)	7.6 (.3)	**1.54 (.07)**
CelebA noseY	1.64 (.09)	–	2.5 (.2)	12.3 (1.1)	**1.39 (.04)**
Airbnb	0.156 (.002)	–	0.158 (.003)	0.153 (.003)	**0.142 (.002)**

Linear Mixed Models (LMM) (I)

$$\boxed{y} = \boxed{X\beta} + \boxed{Zb} + \boxed{\varepsilon}$$

$y \in R^n$	$n \times p$ “fixed” features matrix	$\beta \in R^p$ “fixed” effects	$n \times q$ “random” features matrix	$b \in R^q$ “random” effects (RE) $\sim N(0, D_{q\times q})$	$\varepsilon \in R^n$ $\sim N(0, \sigma_e^2 I_n)$

Random Intercepts model:

- single categorical (RE) feature of q levels
- $D = \sigma_b^2 I_q$
- Z is a binary matrix
- s.t. $y_{ij} = \beta_0 + \beta_1 x_{ij,1} + \cdots + \beta_1 x_{ij,p-1} + b_j + \varepsilon_{ij}, i = 1, \dots, n_i, j = 1, \dots, q$

Linear Mixed Models (LMM) (II)

- Marginal distribution of y:

$$y \sim N(X\beta, V(\psi))$$

where ψ are *variance components* to estimate and $V(\psi) = ZDZ' + \sigma_e^2 I_n$

- Log-likelihood of β, ψ:

$$l(\beta, \psi) = -\frac{1}{2}(y - X\beta)'V(\psi)^{-1}(y - X\beta) - \frac{1}{2}\log|V(\psi)| - \frac{n}{2}\log 2\pi$$

- Get $\hat{\beta}, \hat{\psi}$ via MLE/REML
- Predict (BLUP) $\hat{b} = DZ'V\left(\hat{\psi}\right)^{-1}(y - X\hat{\beta})$ (avoid inversion for random intercepts...)
- Predict $\hat{y}_{te} = X_{te}\hat{\beta} + \mathrm{Z}_{te}\hat{b}$ or $\hat{y}_{te} = X_{te}\hat{\beta}$ for unknown levels

Random effects: what for

By treating high-cardinality categorical features as RE in DNN, we hope to:

- Model the correlation between clustered observations better (faces of the same person!)
- Scale to higher q (only 2 additional params to estimate for random intercepts, many DNN ignore such features altogether)
- Ultimately leading to better prediction performance (e.g. MSE)
- Bonus: scale non-linear MM as well!

Our approach (LMMNN*) (I)

$y = X\beta + Zb + \varepsilon$

$$y = \boxed{f(X)} + \boxed{g(Z)b} + \varepsilon$$

“fixed” part: output of (any) regression DNN

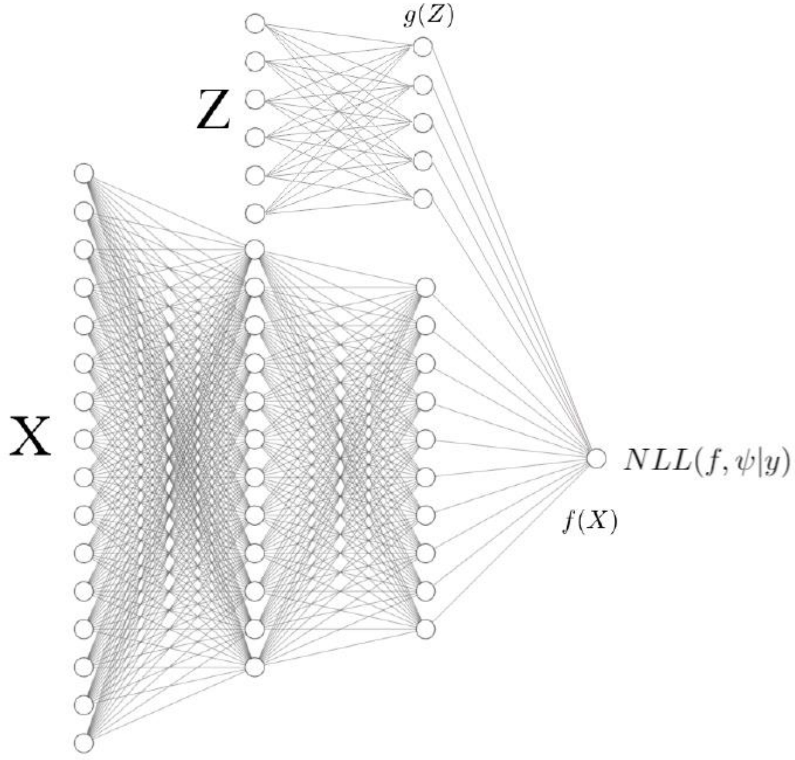
“random” part: output of NN / embeddings, many times $g(Z) = Z$

$b \in R^d$ “random” effects (RE) $\sim N(0, D_{d\times d})$

$\varepsilon \in R^n \sim N(0, \sigma_e^2 I_n)$

*In this work we focus on the random intercepts model for high-cardinality categorical features in a regression setting first, see extensions later.

Our approach (LMMNN) (II)



Our approach (LMMNN) (III)

- Marginal distribution of y:

$$y \sim N(f(X), V(g,\psi))$$

where ψ are *variance components* to estimate and $V(g,\psi) = g(Z)Dg(Z)' + \sigma_e^2 I_n$

- *Negative* log-likelihood (NLL) of f, g, ψ is a natural loss function at each gradient descent iteration on mini-batch ξ of size m:

$$\mathrm{NLL}_\xi(f,g,\psi) = \frac{1}{2}\left(y_\xi - f(X_\xi)\right)' V(g,\psi)_\xi^{-1}\left(y_\xi - f(X_\xi)\right) + \frac{1}{2}\log\left|V(g,\psi)_\xi\right| + \frac{m}{2}\log 2\pi$$

- Get $\hat{f}, \hat{g}, \hat{\psi}$ via DNN optimization (mini-batch SGD, auto-differentiation)
- Predict (BLUP) $\hat{b} = DZ'V\left(\hat{g},\hat{\psi}\right)^{-1}(y - \hat{f}(X))$ (avoid inversion for random intercepts…)
- Predict $\hat{y}_{te} = \hat{f}(X_{te}) + \hat{g}(Z_{te})\hat{b}$ or $\hat{y}_{te} = \hat{f}(X_{te})$for unknown levels

Other approaches

- MeNets (Xiong et. al., 2019)

$$y = f(X)\beta + f(Z)b + \varepsilon$$

- DeepGLMM (Tran et. al., 2020)

$$g(\mu_{it}) = f\left(x_{it}^{(1)}, \omega, \beta^{(1)}\right) + (\beta^{(2)} + \alpha_i)' x_{it}^{(2)}$$

Results: Simulated data (I)

- $y = (X_1 + \cdots + X_{10}) \cdot \cos(X_1 + \cdots + X_{10}) + 2 \cdot X_1 \cdot X_2 + g(Z)b + \varepsilon$
- $X_l \sim U(-1, 1); l = 1, \ldots, 10$
- $\sigma_e^2 = 1$
- $q \in \{100, 1{,}000, 10{,}000\}; \sigma_b^2 \in \{0.1, 1.0, 10.0\}$
- $n = 100{,}000, 80/20\%$ train/test split
- 5 repetitions
- Base DNN: 4-layers MLP with [100, 50, 25, 12, 1] neurons, 25% Dropout, ReLU activation

Results: Simulated data (II)

Table 1: Simulated model with $g(Z) = Z$, mean test MSEs and standard errors in parentheses. Bold results are non-inferior to the best result in a paired t-test.

σ_b^2	q	Ignore	OHE	Embeddings	lme4	MeNets	LMMNN
0.1	10^2	1.25 (.012)	1.20 (.010)	1.18 (.006)	2.92 (.017)	1.15 (.013)	**1.14 (.010)**
	10^3	1.23 (.009)	1.31 (.008)	1.24 (.004)	2.96 (.022)	1.40 (.065)	**1.14 (.009)**
	10^4	1.22 (.004)	1.54 (.008)	1.56 (.007)	2.97 (.014)	1.51 (.133)	**1.17 (.010)**
1	10^2	2.17 (.041)	1.23 (.008)	1.21 (.010)	2.93 (.013)	1.22 (.022)	**1.09 (.010)**
	10^3	2.16 (.015)	1.39 (.015)	1.32 (.014)	2.94 (.013)	1.42 (.091)	**1.14 (.006)**
	10^4	2.14 (.013)	1.68 (.013)	1.68 (.013)	3.17 (.021)	1.66 (.056)	**1.27 (.014)**
10	10^2	10.45 (.38)	1.56 (.044)	1.57 (.039)	2.92 (.012)	1.86 (.156)	**1.10 (.013)**
	10^3	11.37 (.11)	1.75 (.022)	1.72 (.041)	2.95 (.024)	2.19 (.143)	**1.12 (.015)**
	10^4	11.31 (.04)	2.34 (.027)	2.20 (.033)	3.37 (.020)	3.29 (.423)	**1.32 (.007)**

Results: Simulated data (III)

Table 3: Simulated model with $\boxed{g(Z) = ZW}$, mean test MSEs and standard errors in parentheses.

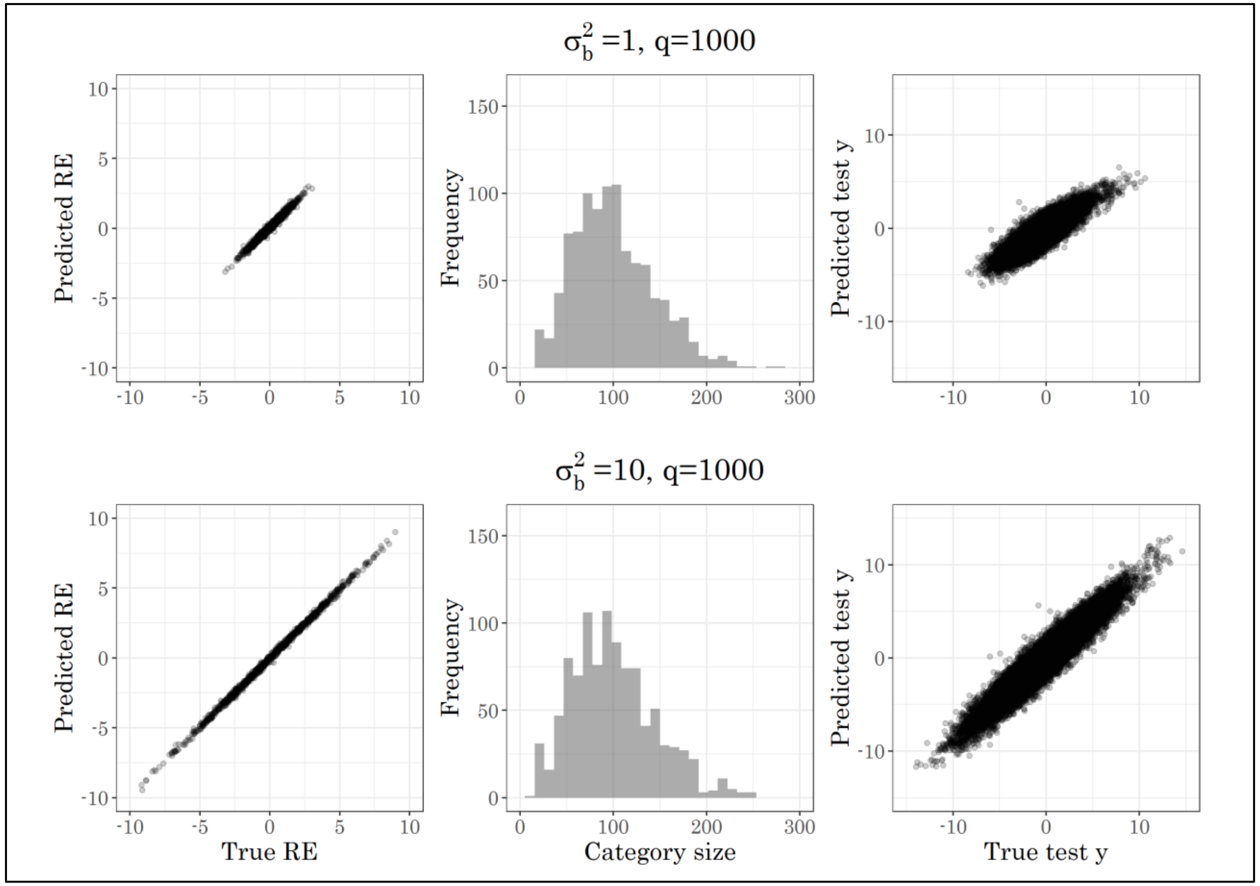
σ_b^2	q	Ignore	OHE	Embeddings	lme4	MeNets	LMMNN
0.1	10^2	1.42 (.039)	1.22 (.018)	1.19 (.024)	2.91 (.021)	1.25 (.084)	**1.13 (.013)**
	10^3	4.92 (.345)	1.49 (.033)	1.43 (.035)	2.95 (.020)	1.44 (.061)	**1.16 (.013)**
	10^4	35.1 (.456)	3.25 (.086)	3.39 (.116)	3.42 (.036)	7.35 (1.9)	**1.59 (.023)**
1	10^2	4.13 (.626)	1.32 (.025)	1.31 (.033)	2.88 (.023)	1.40 (.106)	**1.16 (.011)**
	10^3	35.6 (2.78)	2.49 (.151)	2.56 (.355)	2.96 (.045)	7.00 (1.9)	**1.19 (.028)**
	10^4	334 (18)	9.13 (2.29)	14.4 (2.89)	4.29 (.1)	143.3 (32)	**3.43 (.757)**
10	10^2	32.5 (6.86)	1.74 (.121)	2.43 (.317)	2.90 (.023)	12.0 (3.03)	**1.12 (.012)**
	10^3	324 (25)	8.94 (.79)	9.27 (.92)	2.96 (.031)	164 (18)	**1.20 (.020)**
	10^4	3337 (134)	60.1 (4.6)	91.1 (4.5)	**13.8 (1.2)**	2880 (463)	**13.3 (2.0)**

Results: Simulated data (IV)

Table 2: Simulated model, estimated variance components on average

		$g(Z) = Z$				$g(Z) = ZW$			
		lme4		LMMNN		lme4		LMMNN	
σ_b^2	q	$\hat{\sigma}_e^2$	$\hat{\sigma}_b^2$	$\hat{\sigma}_e^2$	$\hat{\sigma}_b^2$	$\hat{\sigma}_e^2$	$\hat{\sigma}_b^2$	$\hat{\sigma}_e^2$	$\hat{\sigma}_b^2$
0.1	10^2	2.92	0.09	1.12	0.10	2.92	0.49	1.09	0.15
	10^3	2.91	0.10	1.16	0.10	2.91	3.52	0.92	0.16
	10^4	2.90	0.10	1.17	0.17	2.91	33.8	0.19	0.16
1	10^2	2.90	1.01	1.12	1.00	2.91	2.44	1.07	0.41
	10^3	2.90	0.98	1.15	1.00	2.90	32.0	0.84	0.43
	10^4	2.90	0.99	1.26	1.02	2.92	336.6	0.19	0.35
10	10^2	2.90	10.13	1.04	9.24	2.89	32.9	1.06	2.21
	10^3	2.91	10.02	1.12	10.01	2.89	337.8	0.75	1.30
	10^4	2.91	10.01	1.34	9.72	2.90	3305.6	0.22	1.98

Results: Simulated data (V)



Results: Real data

Table 4: Real data features summary table

Dataset	n	q	p	Categorical	y	Input Type	DNN
UKB PA	96K	350	15	job	PA	Tabular	MLP
Drugs	215K	3.6K	10K	drug	rating	Text	LSTM
CelebA	202K	10K	218x178x3	identity	noseX-Y	Images	CNN
Airbnb	50K	40K	196	host	log(price)	Tabular	MLP

Table 5: Real data 5-CV mean test MSEs and standard errors in parentheses.

Dataset	Ignore	OHE	Embeddings	MeNets	LMMNN
UKB PA	**0.812 (.008)**	0.816 (.009)	0.817 (.010)	**0.811 (.009)**	**0.809 (.008)**
Drugs	2.74 (.032)	2.77 (.005)	2.72 (.051)	2.81 (.031)	**2.66 (.006)**
CelebA noseX	**1.68 (.05)**	–	3.6 (.3)	7.6 (.3)	**1.54 (.07)**
CelebA noseY	1.64 (.09)	–	2.5 (.2)	12.3 (1.1)	**1.39 (.04)**
Airbnb	0.156 (.002)	–	0.158 (.003)	0.153 (.003)	**0.142 (.002)**

Convergence

- Main challenge in decomposing the full NLL gradient: $V(g,\psi)^{-1}$and $\log|V(g,\psi)|$ where V is $n \times n$ and only a $m \times m$ "part" of it is used in each mini-batch of size m
- For the simple random intercepts model with $g(Z) = Z$ we show in the paper how the full gradient can be written exactly as the sum of q sub-gradients where each batch consists of n_j level j observations:

$$\frac{\partial NLL}{\partial \psi} = \sum_{j=1}^{q}\left[-\frac{1}{2}\left(y_j - f(X_j)\right)' V_j^{-1}\frac{\partial V_j}{\partial \psi} V_j^{-1}\left(y_j - f(X_j)\right) + \frac{1}{2} tr\left(V_j^{-1}\frac{\partial V_j}{\partial \psi}\right)\right]$$

- This:
  - may not be realistic for some datasets as n_j could be large for some j
  - is not the case in general when $g(Z) \neq Z$
  - for other LMM scenarios, needs a block-diagonal $V(g,\psi)$ (see Extensions)
- For a more general structure of $V(g,\psi)$ see e.g. Chen et. al. 2020, but this is still WIP

Extensions

- A single high-cardinality categorical feature is just the start
- Many useful correlation structures D have already shown promising results (future work):
  - Multiple categorical features (see simulations)
  - Longitudinal data
  - Kriging over random fields, similar to GP
  - GLMM (e.g. classification setting)

Table 6: Simulated model with $g(Z) = Z$ and two categorical features, mean test MSEs and standard errors in parentheses.

σ^2_{b1}	σ^2_{b2}	q_1	q_2	Ignore	OHE	Embeddings	lme4	LMMNN
0.5	0.5	10^3	10^3	2.18 (.03)	1.45 (.02)	1.34 (.01)	2.97 (.03)	**1.13 (.01)**
		10^3	10^4	2.15 (.02)	1.70 (.01)	1.68 (.02)	3.12 (.03)	**1.23 (.01)**
		10^4	10^4	2.13 (.02)	1.83 (.02)	1.80 (.02)	3.23 (.03)	**1.30 (.00)**
0.5	5.0	10^3	10^3	6.73 (.04)	1.66 (.03)	1.57 (.02)	3.00 (.04)	**1.12 (.00)**
		10^3	10^4	6.75 (.04)	2.20 (.03)	2.01 (.03)	3.31 (.02)	**1.29 (.01)**
		10^4	10^3	6.50 (.05)	1.88 (.03)	1.92 (.04)	3.15 (.01)	**1.23 (.01)**
		10^4	10^4	6.68 (.12)	2.48 (.03)	2.16 (.02)	3.43 (.01)	**1.37 (.01)**
5.0	5.0	10^3	10^3	11.26 (.19)	1.83 (.02)	1.80 (.07)	2.97 (.02)	**1.14 (.02)**
		10^3	10^4	11.33 (.19)	2.36 (.03)	2.11 (.02)	3.32 (.02)	**1.30 (.01)**
		10^4	10^4	11.24 (.09)	3.02 (.03)	2.55 (.000)	3.69 (.02)	**1.49 (.02)**

Code

- Python 3.8, Tensorflow-Keras (Chollet, 2015)
- `lmmnn` package, key feature: `NLL` custom Keras loss layer
- All code in: https://github.com/gsimchoni/lmmnn

```python
# after importing all that is necessary for Keras
from lmmnn.layers import NLL

def cnn_lmmnn():
    input_layer = Input((IMG_HEIGHT, IMG_WIDTH, 3))
    y_true_input = Input(shape=(1, ),)
    Z_input = Input(shape=(1, ), dtype=tf.int64)
    x = Conv2D(64, (2, 2), activation='relu')(input_layer)
    x = MaxPool2D((2, 2))(x)
    x = Conv2D(32, (2, 2), activation='relu')(x)
    x = MaxPool2D((2, 2))(x)
    x = Conv2D(16, (2, 2), activation='relu')(x)
    x = MaxPool2D((2, 2))(x)
    x = Flatten()(x)
    x = Dense(100, activation='relu')(x)
    y_pred_output = Dense(1)(x)
    nll = NLL('intercepts', 1.0, [1.0])(
        y_true_input, y_pred_output, [Z_input]
    )
    return Model(
        inputs = [input_layer, y_true_input, Z_input],
        outputs=nll
    )

model = cnn_lmmnn()
model.compile(optimizer= 'adam')

history = model.fit(
    [X_train['images'], y_train, X_train[RE_feature]],
    None,
    validation_split = 0.1,
    batch_size = batch_size, epochs=epochs
)
```

References

- Yu-Wei Lin, Yuqian Zhou, Faraz Faghri, Michael J. Shaw, and Roy H. Campbell. Analysis and prediction of unplanned intensive care unit readmission using recurrent neural networks with long short-term memory. PLoS ONE, 14(7), July 2019. doi: 10.1371/journal.pone.0218942. URL.
- Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015
- Pouya Rezazadeh Kalehbasti, Liubov Nikolenko, and Hoormazd Rezaei. Airbnb price prediction using machine learning and sentiment analysis, 2019
- John T. Hancock and Taghi M. Khoshgoftaar. Survey on categorical data for neural networks. Journal of Big Data, 7(1):28, Apr 2020. ISSN 2196-1115. doi: 10.1186/s40537-020-00305-w. URL.
- Hao Chen, Lili Zheng, Raed AL Kontar, and Garvesh Raskutti. Stochastic gradient descent in correlated settings: A study on gaussian processes. In H. Larochelle,M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 2722–2733. Curran Associates, Inc., 2020. URL.
- François Chollet et al. Keras. https://keras.io, 2015.

Acknowledgements

This study was supported in part by a fellowship from the Edmond J. Safra Center for Bioinformatics at Tel-Aviv University, and by Israeli Science Foundation grant 1804/16. UK Biobank research has been conducted using the UK Biobank Resource under Application Number 56885.

Thank you.