Supplementary Material

1.1) Comparison of major neural cell types

Methods

First, we wanted to establish the viability of cross-species comparisons at the major cell type level for neurons and glia. We used the MusNG and HumN datasets. We identified a 439 gene feature set using maximum relevance minimum redundancy (Ding and Peng, 2005) in mouse to differentiate the major cell types located in the brain (interneuron, S1 pyramidal, CA1 pyramidal, oligodendrocyte, microglia, endothelial, astrocyte, ependymal, mural, as identified by the Linnarson group (Zeisel, et al., 2015)). ependymal and mural cell types were not identified in the HumN dataset and therefore were used as a negative control.

Before the data was submitted to mRMR, we performed feature selection (Zeisel, et al., 2015) consisting of i) removing low expression genes, ii) removing genes that do not coexpress with other genes and iii) retaining the top 5000 high variance genes using a noise model fit to the data (Zeisel, et al., 2015). The samples were separated into the 9 major cell types with the 5000 highest variance genes. This labeled dataset was submitted to the mRMR algorithm resulting in 500 optimally orthogonal and high variance genes. After filtering genes that matched between the MusNG and HumNG datasets, the 500 mRMR-selected genes were further reduced to 439.

These genes were then used to train a NN classifier on the MusNG data separated into 3 groups, train (60%), validation (20%) and test (20%). The model was trained using 10 L_2 regularization values (Lambda: 0.0, 0.001, 0.01, 0.05, 0.2, 0.5, 1, 2, 5 10, Fig 4) on the training set, and the final lambda value was chosen by selecting the lambda value with the highest overall accuracy on the validation set. The resulting highest accuracy NN model was used to predict cell types in the MusNG test set and in the entire HumNG dataset (Darmanis, et al., 2015).
As a comparison, a regularized logistic regression (LR) classifier was trained on each of the cell types using the same 439 gene feature set. The training set consisted of the same samples as the NN classifier. The validation set and test set were also the same NN samples. LR was conducted using the same lambda values as the NN classifier resulting 90 models, 9 for each cell type for 10 different lambda values. The best performing lambda (across all cell types) selected by overall accuracy on the test set was used for all 9 models and evaluated on the MusNG validation set and the entire HumNG dataset. We evaluated the accuracy of these models using overall accuracy $\left(\frac{TP}{(TP+FP+TN+FN)}\right)$ and weighted accuracy $\left(\sum_{i=1}^{|\{major\ cell\ types\}|}\frac{TP_i}{|\{major\ cell\ types\}|\times(TP_i+FP_i+TN_i+FN_i)}\right)$ across each cell type ($i \in major\ cell\ types$).

Results

We observed that the regularized neural network (NN) model performed better than the regularized logistic regression (LR) model in the mouse dataset (overall accuracy: LR = 0.87 and NN = 0.91 respectively, **Supplementary Fig. S1**). Moreover, the overall accuracy increased greatly when comparing across species (overall accuracy: LR = 0.4 and NN = 0.55, **Supplementary Fig. S1**). We also see the negative controls (ependymal and mural, I.e., no correspondence in the human dataset) have few cells assigned to them when using the NN (**Supplementary Fig. S1D**).

Aside from the overall accuracy, which was biased by the number of cells in each cell type, we wanted to view the weighted accuracy to get a representation of the performance across cell types. We found that the NN classifier performed better in the mouse validation set (weighted accuracy: LR = 0.69 and NN = 0.74, **Supplementary Fig. S1**). More intriguingly, the human dataset weighted accuracy was even more improved (weighted accuracy: LR = 0.46 and NN = 0.61, **Supplementary Fig. S1**).

These results showed that the gene feature set was useful between species but could be improved to provide more accurate classification. There are clear advantages of using *a priori* feature sets to identify cell type similarity (Crow, et al., 2018) but we also aim to explore how feature reduction through a neural network could identify corresponding neural subtypes.

1.2) Comparison of neural subtypes

Methods

Next, we wanted to explore the more granular interneuron and pyramidal subtypes available to us in HumN and MusNG datasets. The mouse dataset constituted both neural and glial cell types while the human dataset constituted only neural cell types. This lack of cell types in human provided us with a negative control. We needed to retain more features due to the increased granularity of the subtypes (Supplementary Material Sec 1.2) vs. major cell types (Supplementary Material Sec 1.1). First, we identified the overlapping gene feature set between both datasets (13355 genes). Next, we converted the cell subtypes into numeric form so that a model could be trained. Because the NN model performed better than a logistic regression classifier in our previous experiment, we only trained NN classifiers. We performed 50 iterations random cross validation on mouse NN classifiers (80% training and 20% testing) using all 48 cell type labels. This process was repeated for multiple lambda regularization values (0.0, 0.001, 0.01, 0.1, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5). These multiple classifiers corresponded to the multiple lambda regularization values that we wanted to test. This model then was used classify MusNG test set and entire HumN dataset.

Because the major cell types are known in human and mouse, we recorded the accuracy in identifying the major cell types. The same process was repeated for the human dataset in which a model was trained on 80% of the human dataset and tested on 20% 50 times. Because the human dataset only contained neural cell types, only mouse neuron subtypes were used in the human-trained NN classifier. For both datasets, the major type accuracy was recorded for both mouse and human test sets. The subtype level accuracy was recorded for the within species test set.

Results

We observed that the major divisions of interneurons and pyramidal cells were conserved across species. The mouse trained model achieved mouse subtype accuracy of 67% and mouse major cell type accuracy of 93% (**Supplementary Fig. S2A,C**). The mouse trained model was also able to accurately identify major cell types in human cells (**Supplementary Fig. S2A,C**): i) human interneurons were classified interneuron (anova p-value=8.17×10^{-9}), ii) human pyramidal cells were classified pyramidal (anova p-value=4.87×10^{-6}), iii) human cells classified interneuron were interneuron (anova p-value=1.06×10^{-6}), and iv) human cells classified pyramidal were pyramidal cells (anova p-value=1.14×10^{-2}). We saw similarly high levels of accuracy in human.

The human trained model achieved human subtype accuracy of 94% and human major cell type accuracy of 99% (**Supplementary Fig. S2B,D**, note: the human dataset contained fewer subtypes and major cell types increasing the accuracy). The human trained model was also able to accurately identify major cell types in mouse cells (**Supplementary Fig. S2B,D**): i) mouse interneurons were classified interneuron (anova p-value=6.00×10^{-11}), ii) mouse pyramidal cells were classified pyramidal (anova p-value=1.55×10^{-2}), iii) mouse cells classified interneuron were interneuron (anova p-value=3.75×10^{-4}), and iv) mouse cells classified pyramidal were pyramidal cells (anova p-value=3.38×10^{-3}).

In human, the cell types were rarely associated with a single mouse cell type but rather related to combinations of mouse subtypes. For example, human cell type: In6(6) was related to mouse cell type: Int2(2) and Int14(14) in both mouse and human trained models. Human cell type: In1(1) was related to mouse cell type: Int12(12) (**Fig. 3**). Aside from the interneuron cells, we saw relationships in the pyramidal cell types as well. Human cell type: Ex1(9) was related to mouse cell type: S1PyrL23(17) in both human and mouse trained models. Human cell type: Ex3(11) was related to mouse cell type: S1PyrL4(18) in both human and mouse trained models. These relationships advance the idea that these tissue specific subtype profiles are conserved across species and can be leveraged to identify conserved neural subtypes (**Supplementary Fig. S2**).

Also, we found that the cortical layer specific subtypes were mirrored in the classification between species. This can most prominently be seen in **Supplementary Fig. S2C** where human and mouse cells from the same cortical layers correspond to one another (i.e., Ex1(9)→S1PyrL23(17), Ex2(10)→S1PyrL4(18), Ex3(11)→S1PyrL4(18), Ex4(12)→S1PyrL4(18), Ex5(13)→S1PyrL5(19), Ex6→S1PyrL5(19)/S1PyrL6(20)).

2.1) Detailed Implementation

Data Preprocessing

To follow the algorithms above, we convert the general framework above into a concrete implementation. The possible label mappings of the original subtypes to the conserved subtypes are established from the literature (Darmanis, et al., 2015; Lake, et al., 2016; Zeisel, et al., 2015). We determine the known relationships between L and $\hat{L}$ and represent them using the label mask ($\boldsymbol{G}$ in **Eq S1**). A subtype L_i can be exactly the same subtype as $\hat{L}_j$ (e.g., $MusNG\ Interneuron\ 5\ \equiv MusNG\ Interneuron\ 5$), potentially the same subtype as $\hat{L}_j$ (e.g., $MusNG\ Interneuron\ 5\ \cong HumN\ Interneuron\ 2$), or not the same subtype as $\hat{L}_j$ (e.g., $MusNG\ Interneuron\ 5\ \neq HumNG\ Oligodendrocyte$).

$$\boldsymbol{G} \in \mathbb{Z}^{l \times \hat{l}}$$

$$\boldsymbol{G_{i,j}} = \begin{cases} 1, & L_i \equiv \widehat{L_j} \\ 1, & L_i \cong \widehat{L_j} \\ 0, & L_i \neq \widehat{L_j} \end{cases} \quad \text{Eq S1}$$

Once the proper label mask ($\boldsymbol{G}$) is generated (**Eq S1**) all of the necessary data is available to run LAmbDA. These data must now be processed before performing Algorithm 1 or 2. Firstly, only genes with variance in the top 20th percentile and with fewer than 50% zeros are selected. The entire matrix is divided by 10 and log2 transformed ($log2(\boldsymbol{X}/10+1)$). Next, sample imbalances are addressed by over-sampling under-represented cell subtypes and under-sampling the over-represented cell subtypes. Note that our testing set during cross-validation is removed at this point. A cutoff (c, **Eq S2**) is selected based on a percentile of the label distribution for over and under sampling. The quartile is selected using the optunity (Claesen, et al., 2014) hyperparameter search tool. Optunity searches distributions of hyperparameters based on a cost function provided by the user and returns the set of hyperparameters that minimize the cost function. The resampling is also weighted by the ambiguity of the subtypes (i.e., number of possible labels for a subtype) using a constant (γ). Note that if γ is set to 0 then there will be no weighting based on the label ambiguity.

$$c = percentile(colSums(\boldsymbol{Y}), p_{cut})$$

$$o = c\left\lceil \log_2\left(\left(\frac{\max\left(rowSums(\boldsymbol{G})\right)}{rowSums(\boldsymbol{G})}\right)+1\right)^{\gamma}\right\rceil \quad \text{Eq S2}$$

Variable A is the set of each individual row in $\boldsymbol{Y}$. The combined over/under-sampling t cutoff point (o) results in a new set of indices $\ddot{A}$. The new set of sample indices ($\ddot{A}$) were then used to index our original data matrix ($\boldsymbol{X}$) and label matrix ($\boldsymbol{Y}$) to produce our final data matrix ($\ddot{\boldsymbol{X}}$) and our final label matrix ($\ddot{\boldsymbol{Y}}$, **Eq S3**).

$$\ddot{\boldsymbol{X}} = \boldsymbol{X}_{(:),\ddot{A}}\,, \quad \ddot{\boldsymbol{Y}} = \boldsymbol{Y}_{(:),\ddot{A}},\ \ddot{n} = |\ddot{A}| \quad \text{Eq S3}$$

Assigning labels to ambiguously labeled cells

Due to systematic bias in the datasets, we introduce a dispersion constant (τ) to prevent subtypes from being selected in early iterations and reinforced at a local minima. The dispersion constant is selected using the optunity (Claesen, et al., 2014). To prevent any weighting, the dispersion constant can be set to 0. The dispersion-weighted (d) matrix is denoted $\hat{f}(\ddot{\boldsymbol{X}})^{(d)}$ (**Eq S4**). The constant of ε (=0.1) is added to prevent artificial masking of desired labels.

$$\hat{f}(\ddot{\boldsymbol{X}})^{(d)}_{(i),j,:} = \left(\hat{f}(\ddot{\boldsymbol{X}}_{(i)})+\varepsilon\right)_{(i),j,:}\left[\frac{mean(\hat{f}(\ddot{\boldsymbol{X}}_{(i)})+\varepsilon)}{colMeans(\hat{f}(\ddot{\boldsymbol{X}}_{(i)})+\varepsilon)}\right]^{\tau} \quad \text{Eq S4}$$

$$|\ i,j \in \mathbb{Z}, 1 \le i \le 3, 1 \le j \le \ddot{n}$$

Then, the label mask ($\boldsymbol{G}$) is used to set all incorrect labels to 0 so that they will not be selected as the correct label. Note that "∘" denotes element-wise multiplication (e.g., Hadamard product). The masked (m) dispersion-weighted (d) matrix is denoted $\hat{f}(\ddot{\boldsymbol{X}})^{(m)(d)}$ (**Eq S5**).

$$\hat{f}(\ddot{\boldsymbol{X}})^{(m)(d)} = \ddot{\boldsymbol{Y}}\boldsymbol{G} \circ \hat{f}(\ddot{\boldsymbol{X}})^{(d)} \quad \text{Eq S5}$$

Each of the original subtype labels in matrix L should be grouped with similar subtypes $\hat{L}$. To enforce this, a subtype weight is added, resulting in the subtype weight matrix ($\hat{f}(\ddot{\boldsymbol{X}})^{(s)}$) (**Eq S6**).

$$\hat{f}(\ddot{\boldsymbol{X}})^{(s)} = \ddot{\boldsymbol{Y}}\ddot{\boldsymbol{Y}}^{\mathbf{T}}\hat{f}(\ddot{\boldsymbol{X}})^{(m)(d)} \quad \text{Eq S6}$$

The subtype label weights then are combined with the masked dispersion weighted matrix at a proportion specified by Δ. To prevent any correction to the original subtype weights, Δ can be set to 0 (**Eq S7**).

$$\hat{f}(\ddot{\boldsymbol{X}})^{(s)(m)(d)} = \Delta\hat{f}(\ddot{\boldsymbol{X}})^{(s)} + (1-\Delta)\hat{f}(\ddot{\boldsymbol{X}})^{(m)(d)} \qquad \text{Eq S7}$$

The subtype-weighted masked dispersion weighted output ($\hat{f}(\ddot{\boldsymbol{X}})^{(s)(m)(d)}$) now can be converted into labels using one-hot encoding shown below. It is important to note the labels $\hat{L}$ that have a direct correspondence in L can only be assigned to one member of $\hat{L}$. These labels with only one possible mapping are called unambiguous and used in later analyses to determine algorithm performance.

$$\widehat{\boldsymbol{Y}} = onehot\left(argmax\left(\hat{f}(\ddot{\boldsymbol{X}})^{(s)(m)(d)}\right)\right) \qquad \text{Eq S8}$$

Once **Eq S8** is calculated, only function definitions ($\hat{f}(\ddot{\boldsymbol{X}})$) and the loss functions are needed to perform Algorithm 1 for LR and RF. For the NN-based models, the batch effects are removed in the final hidden layer before the subtype predictions are made.

Calculating Batch Effect Error

To remove the batch effects introduced by using multiple datasets we use Euclidean distance between subtypes to reduce inter-dataset distances. In this section, a one hidden layer example is used but can be changed to fit other NN architectures by adding additional layers ($\Theta_{(i)}$). The hidden layer (Θ) is a regularized dropout layer with sigmoid activation function and n_{hidden} hidden nodes. The drop out percentage (p_{drop}) is optimized using sobol search from the optunity package (Claesen, et al., 2014). Dropout is used in the hidden layer(s) because it has been shown to be more robust in high random noise datasets (Rodner, et al., 2016) like scRNA-seq data. To reduce the dataset biases, we add the loss term masks $\boldsymbol{M_1}$ (**Eq S11**), and to increase differences within datasets between labels we add $\boldsymbol{M_2}$ (**Eq S12**). These new constraints are added to the hidden layer outputs using ambiguous label centroid (***C*** in **Eq S9**) squared Euclidean distances (***E*** in **Eq S10**) inspired by computer vision research (Wen, et al., 2016). Note that "tril" denotes the lower triangle of a matrix, $\circ$ denotes element-wise multiplication (e.g., Hadamard product) and $\oslash$ denotes element-wise division (e.g., Hadamard division). With the processing steps (i.e., ambiguous label assignment and batch effects removal) defined, the function ($\hat{f}(x)$) for each model type can be defined.

$$\ddot{\boldsymbol{Y}}_{\boldsymbol{colsum}} = \begin{bmatrix} colSums(\ddot{\boldsymbol{Y}})_1 \\ \vdots \\ colSums(\ddot{\boldsymbol{Y}})_{\ddot{n}} \end{bmatrix}$$

$$\boldsymbol{C} = sigmoid(\ddot{\boldsymbol{X}}\boldsymbol{\Theta})\ddot{\boldsymbol{Y}} \oslash \ddot{\boldsymbol{Y}}_{\boldsymbol{colsum}} \qquad \text{Eq S9}$$

$$\boldsymbol{E} = rowSums(\boldsymbol{C}^2)1^{1\times l} + 1^{l\times 1}rowSums(\boldsymbol{C}^2)^T - 2\boldsymbol{C}\boldsymbol{C}^{\boldsymbol{T}} \qquad \text{Eq S10}$$

$$\boldsymbol{M_1} = \frac{(\boldsymbol{G}\boldsymbol{G}^{\boldsymbol{T}}>0)\circ(1^{l\times l}-(\boldsymbol{D}\boldsymbol{D}^{\boldsymbol{T}}>0))}{tril\left(rowSums(\boldsymbol{G})1^{1\times l}\right)+tril\left(rowSums(\boldsymbol{G})1^{1\times l}\right)^T} \qquad \text{Eq S11}$$

$$\boldsymbol{M_2} = \frac{(\boldsymbol{D}\boldsymbol{D}^{\boldsymbol{T}}>0)}{tril\left(rowSums\left((\boldsymbol{D}\boldsymbol{D}^{\boldsymbol{T}}>0)\right)\right)+tril\left(rowSums\left((\boldsymbol{D}\boldsymbol{D}^{\boldsymbol{T}}>0)\right)\right)^T} \qquad \text{Eq S12}$$

Defining Functions for Each Model Type

We study five different machine learning models ($\hat{f}(x)$) on the processed scRNA-seq data ($\ddot{\boldsymbol{X}}$). Specifically, $\hat{f}(\ddot{\boldsymbol{X}})$ is calculated for LR (**Eq S13**), FF1 (**Eq S14**), FF3 (**Eq S15**), RNN1 (**Eq S16**), and RF (**Eq S17**). The output layer (***O***) is regularized with a softmax activation function. The resulting label matrix $\widehat{\boldsymbol{Y}}$ is used to train an individual iteration of our model ($\hat{f}(\ddot{\boldsymbol{X}})$). The process allows for ambiguous labels to change iteratively to another possible label (**Fig 2B**). The resulting process assigns these ambiguously labeled cells to refined subtypes. For FF3 **Eq S15** instead has three different Θ variables nested in the equation symbolizing the three layers (**Eq S16**). Similarly, the recurrent NN implementation has a recurrent layer (***R***) instead of a single Θ layer (**Eq S17**). The RF method uses aggregate probabilities from trees ($\hat{t}(x)$) with the same scRNA-seq data ($\ddot{\boldsymbol{X}}$). With the models defined, the loss terms to be optimized can be described in detail.

$$\hat{f}(\ddot{\boldsymbol{X}}) = softmax(\ddot{\boldsymbol{X}}\boldsymbol{O}) \quad \text{Eq S13}$$

$$\hat{f}(\ddot{\boldsymbol{X}}) = softmax(sigmoid(\ddot{\boldsymbol{X}}\boldsymbol{\Theta})\boldsymbol{O}) \quad \text{Eq S14}$$

$$\hat{f}(\ddot{\boldsymbol{X}}) = softmax(sigmoid(sigmoid(sigmoid(\ddot{\boldsymbol{X}}\boldsymbol{\Theta_1})\boldsymbol{\Theta_2})\boldsymbol{\Theta_3})\boldsymbol{O}) \quad \text{Eq S15}$$

$$\hat{f}(\ddot{\boldsymbol{X}}) = softmax(sigmoid(\ddot{\boldsymbol{X}}\boldsymbol{R})\boldsymbol{O}) \quad \text{Eq S16}$$

$$\hat{f}(\ddot{\boldsymbol{X}}) = \frac{1}{n_{trees}}\sum_{i=1}^{n_{trees}}(\hat{t}(\ddot{\boldsymbol{X}})) \quad \text{Eq S17}$$

Defining Cost Function for Each Model Type

The cost functions for different model types were slightly different based on the model characteristics most specifically the presence of hidden layer(s). Note the sum of $\Theta_{(i)}$ denotes the sum of all weights on all layers. The RF model used the cost function in **Eq S18** due to the lack of an L2 regularization term. The LR model used the cost function in **Eq S19** since there was no hidden layer but still an L2 regularization term. The four types of NN implementations of LAmbDA (FF1, FF1bag, FF3, and RNN1) (Aymeric, 2018) use the cost function in **Eq S20**. Once the cost functions are defined the model can be trained using most machine learning frameworks like TensorFlow. For the bagging method 5 LAmbDA-FF1 models were run in parallel and the output labels averaged across each model.

$$\min\left(mean\left(\sum\left(\widehat{\boldsymbol{Y}} - f(\ddot{\boldsymbol{X}})\right)^2\right)\right) \quad \text{Eq S18}$$

$$\min\left(mean\left(\sum\left(\widehat{\boldsymbol{Y}} - \hat{f}(\ddot{\boldsymbol{X}})\right)^2\right) + \lambda_1\sum\sum(\boldsymbol{\Theta}_{(i)}{}^2)\right) \quad \text{Eq S19}$$

$$min\left(mean\left(\sum\left(\widehat{\boldsymbol{Y}} - \hat{f}(\ddot{\boldsymbol{X}})\right)^2\right) + \lambda_1\sum\sum(\boldsymbol{\Theta}_{(i)}{}^2) + \lambda_2 mean(\boldsymbol{E}\circ\boldsymbol{M_1}) - \lambda_3 mean(\boldsymbol{E}\circ\boldsymbol{M_2})\right) \quad \text{Eq S20}$$

Additional Information for Running the Algorithms

To train the LAmbDA models, we use the AdamOptimizer (Kingma and Ba, 2014) with step size of 0.01 and random mini-batches of size p_{batch} (a percentage of $\ddot{n}$) that were changed every 50 iterations to prevent overfitting the unambiguous labels. We ran each model for 2000 iterations except for the Random Forest model, which was run for 100 iterations. The code is written for GPU-enabled TensorFlow Python3 package. There were some other considerations that needed to be accounted for during training relating to specific models.

Because recurrent NNs use data order in training, the graph structure within the RNA co-expression data is used to sort the genes data matrix ($\boldsymbol{X}$). Using the top 5000 genes with highest variance, the principle components (PCs) are calculated and then the genes are sorted by the first PC. Next, the expression of the top 5000 gene values are sorted into an 100 by 50 matrix before training, where the sorted genes are separated into 100 rows with 50 genes in each row. This procedure allows for a part of the co-expression network structure to be utilized by the RNN.

There are two non-NN based models tested which do not contain hidden layers: LR and RF. LAmbDA-LR uses the same hyperparameters as the NN models except that n_{hidden}, λ_2, and λ_3 are not needed due to a lack of a hidden layer. LAmbDA-RF does not contain n_{hidden}, λ_1, λ_2, and λ_3 but requires other hyperparameters related to the trees in the random forest. These features are number of trees in the forest (n_{trees}) and maximum number of nodes in the trees (n_{max}). In both of these circumstances the necessary hyperparameters are tuned using the sobol solver in the optunity package (Claesen, et al., 2014).

2.2) LAmbDA model output

Figures 3 and 4 in the main text were also generated for the other LAmbDA variants and methods from other groups to compare and contrast the results. The best overall results for LAmbDA were using the LAmbDA-FF1 and LAmbDA-FF1bag variants so those figures are included in the main text. The other variants in general had good unambiguous accuracy but were not as high accuracy as LAmbDA-FF1bag. One important feature of note is that the LAmbDA-RNN1 model does have good characteristics in integrating datasets evenly (**Supplementary Fig. S3-6**) despite the accuracy drawbacks (**Supplementary Fig. S7-10).** The LAmbDA-RF variant has high unambiguous accuracy (**Table 2**) but does not transfer the knowledge well between datasets (**Supplementary Fig. S7-10**). When AUC is used to map labels between datasets LAmbDA-RF more successfully is able to map ambiguous labels between datasets (**Supplementary Table S4**).

2.3) LAmbDA Hyperparameter Optimization

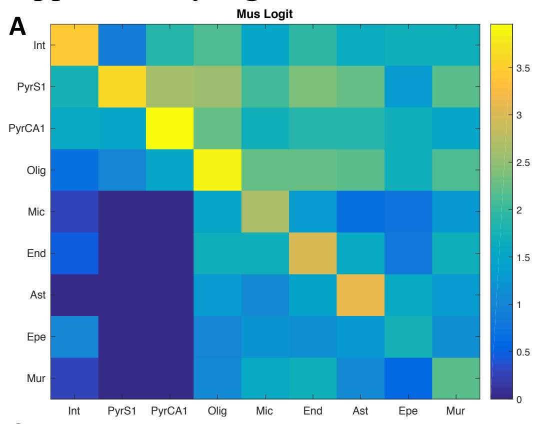
Methods

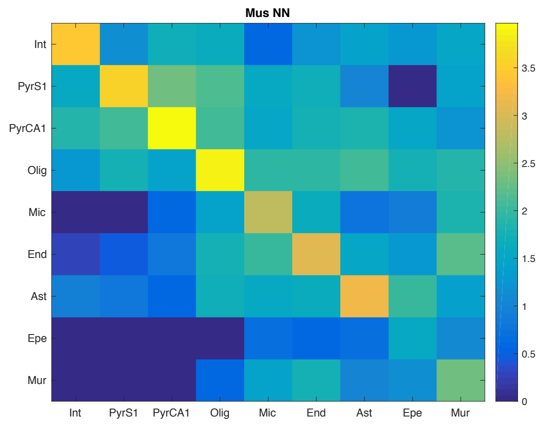
Multiple types of models were trained using the LAmbDA method, which required the tuning of multiple hyperparameters. We tuned these parameters using a sobol solver in the optunity package (Claesen, et al., 2014). During the training, we output these parameters into a file so that the parameters could be studied afterwards. Each algorithm was trained 50 times (10 times for LAmbDA-RNN1) using the sobol-distributed hyperparameters for each iteration of cross validation. The result is that for each of those 500 trained models (100 trained models for LAmbDA-RNN1) we have the error and all of the hyperparameter settings. With these we plot the hyperparameters against the error to better understand how they affect model training.

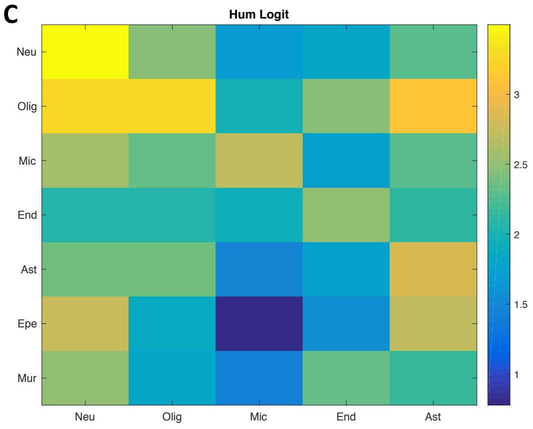
Results

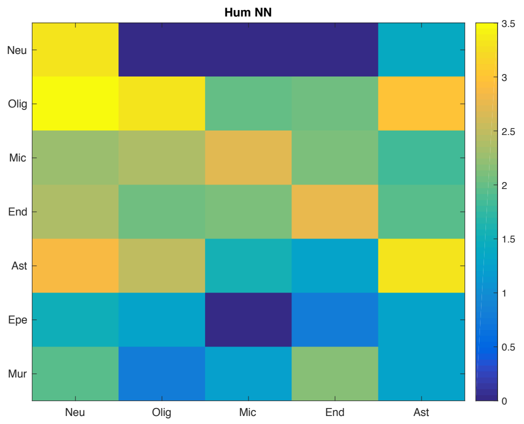
Some of the hyperparameters were more associated with the training error than others. For instance, increased τ tended to increase the error during training (**Supplementary Fig. S11**). However if τ is set too low then many of the subtypes may not be able to disperse properly. Also, higher dropout rate also tended to be associated with higher error. The higher the percentage cutoff also tended to be associated with lower error. Higher γ also tended to be associated with lower error. Though this is seen more drastically in the brain dataset. Another feature of note is that the brain and pancreas dataset though in general mirroring each other's patterns were not identical. This could be due to the higher granularity of the brain dataset.

Supplementary Figures

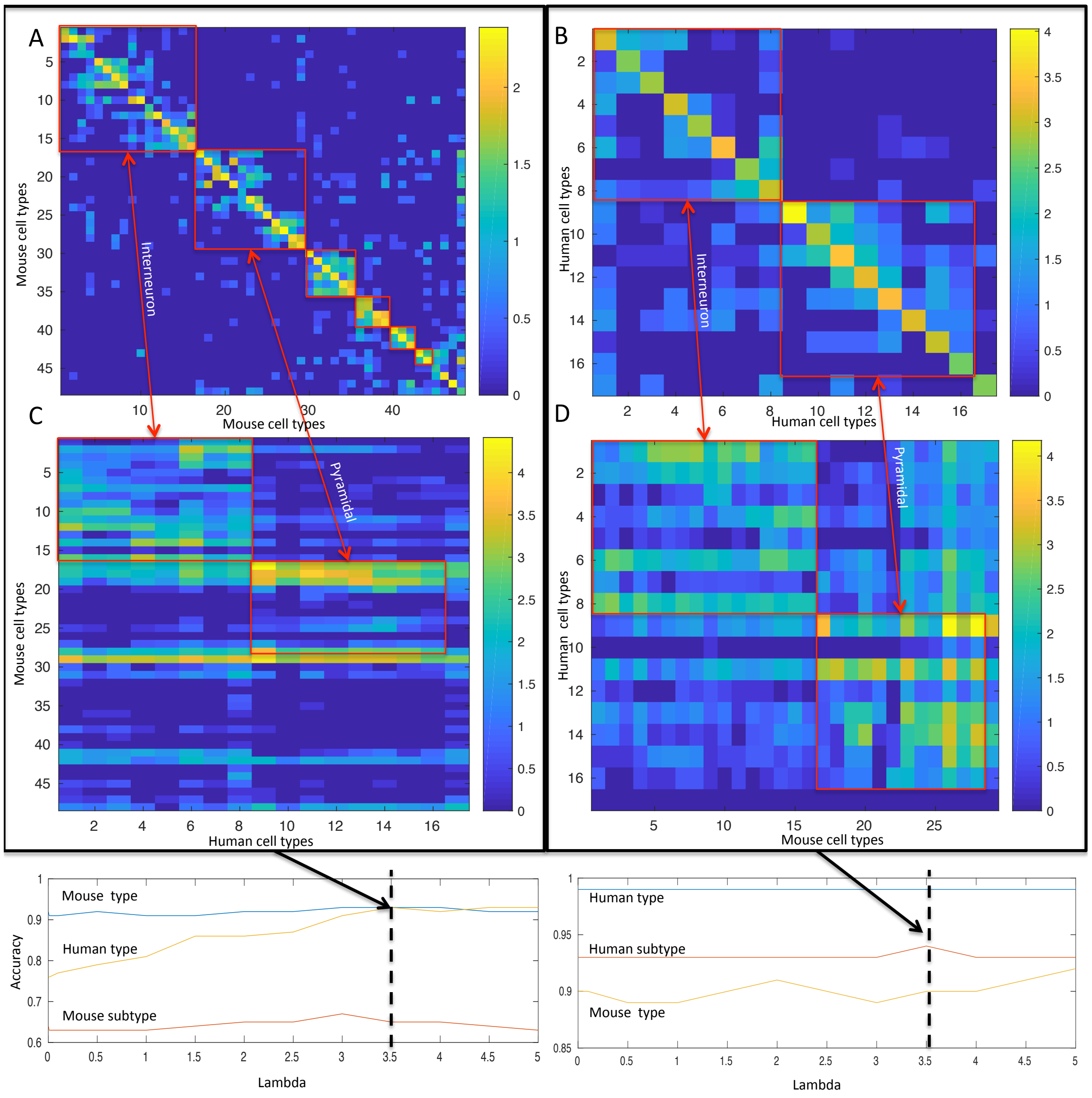
A


B


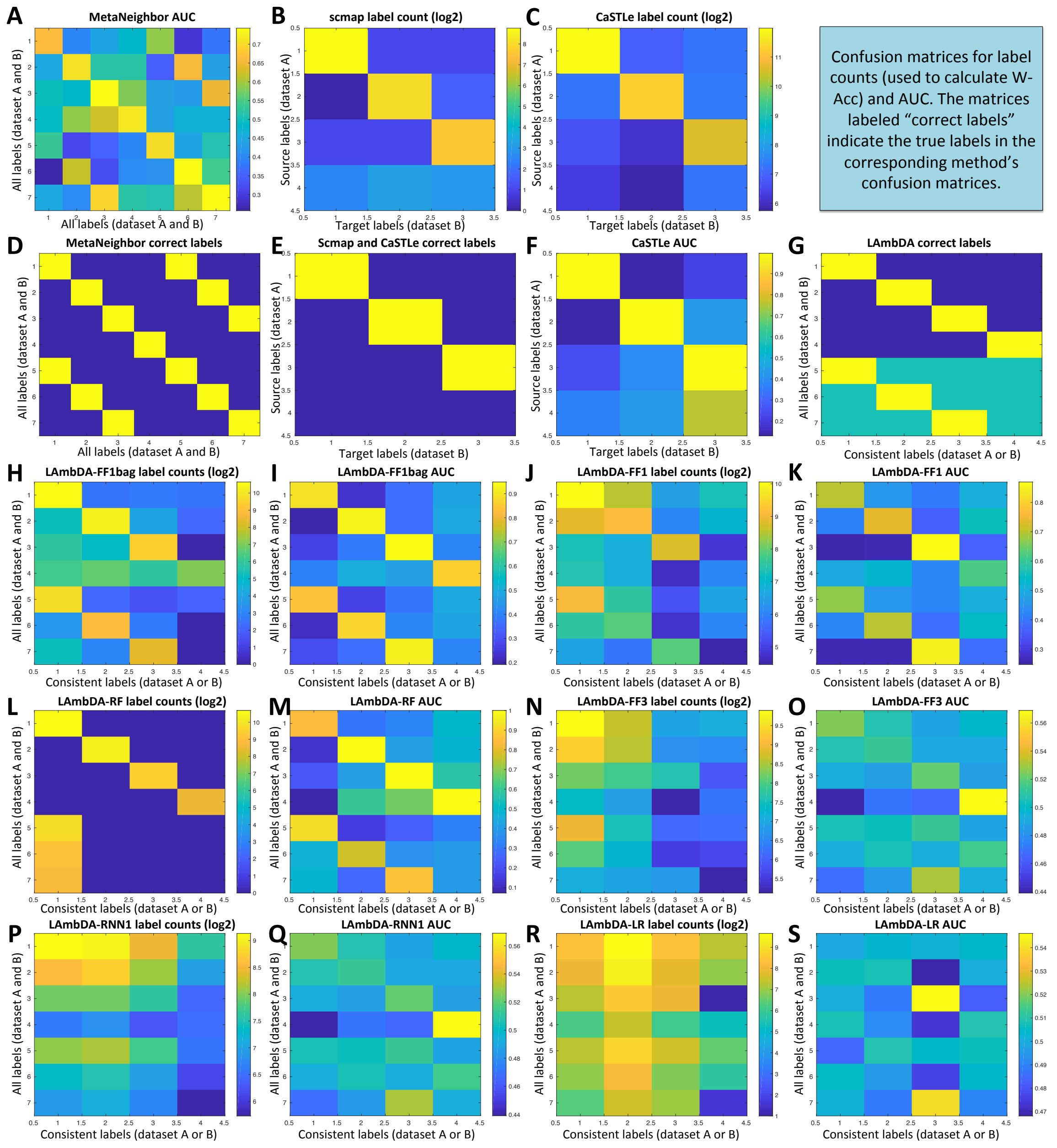
C


D


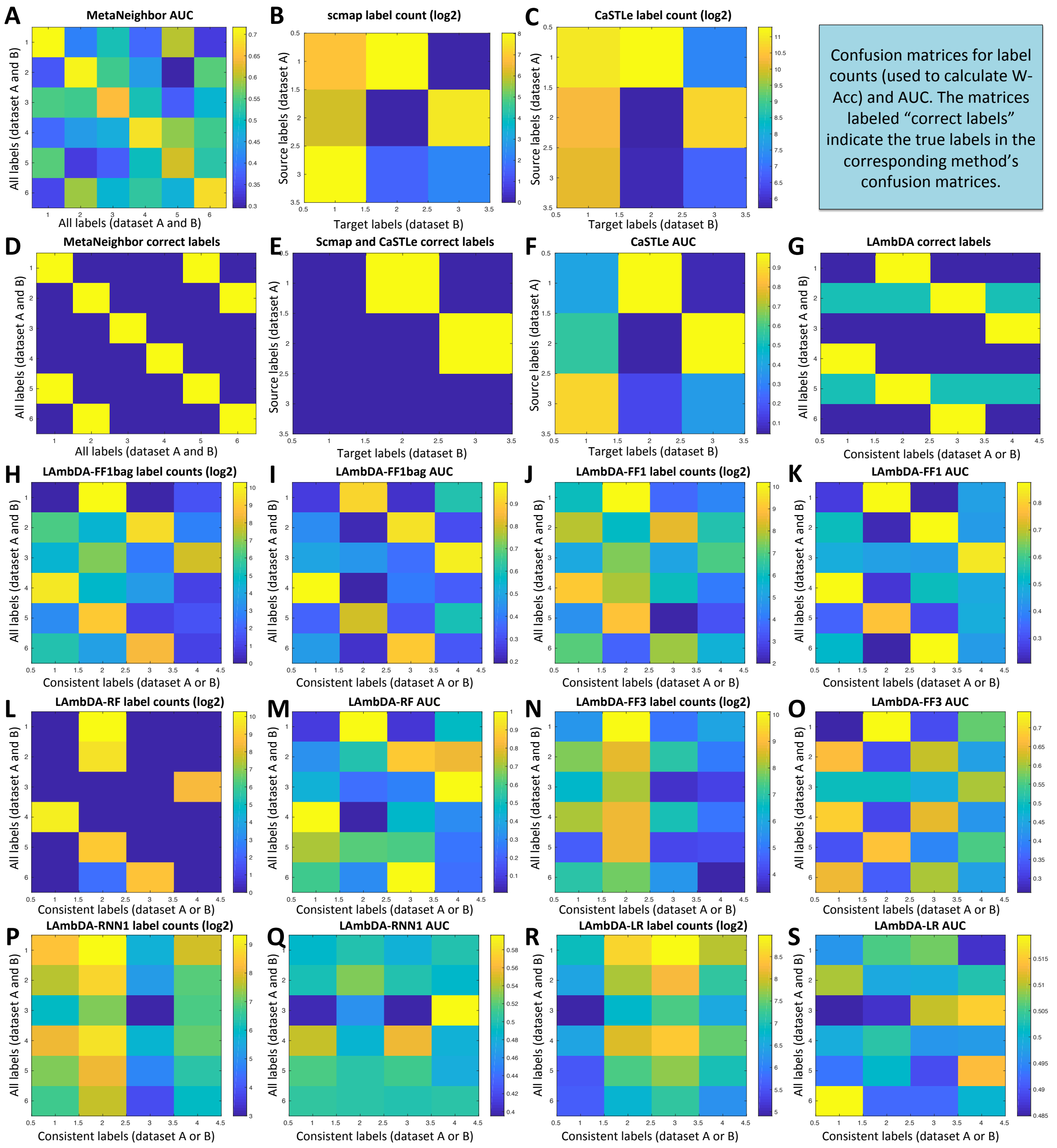
Supplementary Fig. S1. Logit corresponds to the logistic regression model. NN corresponds to the neural network model. For each confusion matrix the rows correspond to predicted major cell types (MusNG) and the columns correspond to true cell types (either MusNG or HumNG). A) The Logit classification confusion matrix (log2) on the MusNG test set (rows: MusNG preds, cols: MusNG labels). B) The NN classification confusion matrix (log2) on the MusNG test set (rows: MusNG preds, cols: MusNG labels). C) The Logit classification confusion matrix (log2) on the HumNG dataset (rows: MusNG preds, cols: HumNG labels). D) The NN classification confusion matrix (log2) on the HumNG dataset (rows: MusNG preds, cols: HumNG labels). Row/Col labels: Neu (neuron), Olig (oligodendrocyte), Mic (microglia), End (endothelial), Ast (astrocyte), Epe (ependymal), and Mur (mural).



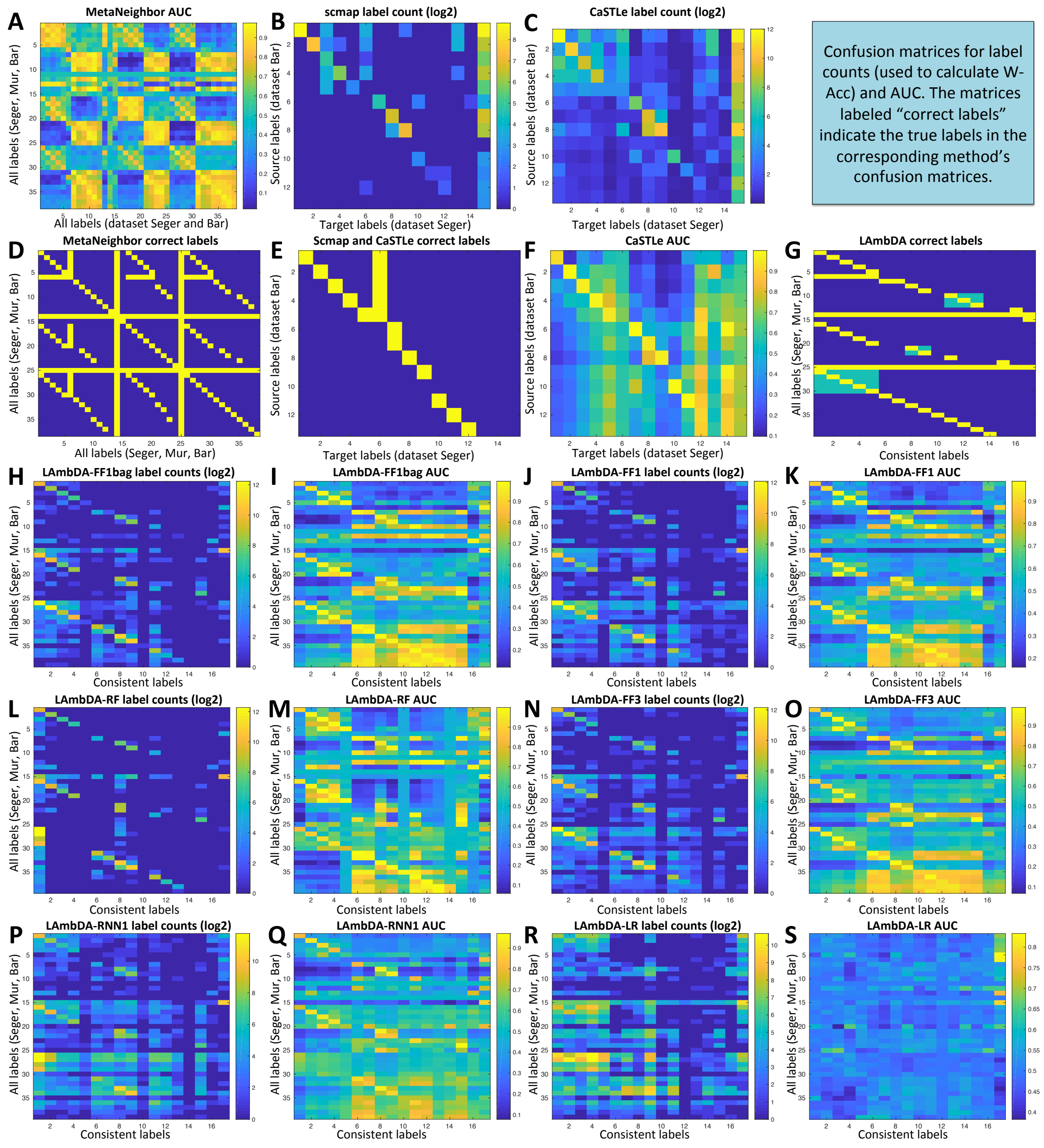
Supplementary Fig. S2. Red boxes indicate major cell types. The two largest red boxes in each panel indicate either interneurons or pyramidal cells. A) The confusion matrix (log2) of the MusNG trained model on the MusNG test set over 50 fold cross valdation at lambda of 3.5. B) The confusion matrix (log2) of the HumN trained model on the HumN test set over 50 fold cross validation at lambda of 3.5. C) The confusion matrix (log2) of the MusNG trained model on the HumN dataset over 50 fold cross validation. D) The confusion matrix (log2) of the HumN trained model on the MusNG dataset over 50 fold cross validation. The bottom panel shows the change in major cell type and subtype accuracy with change in lambda.



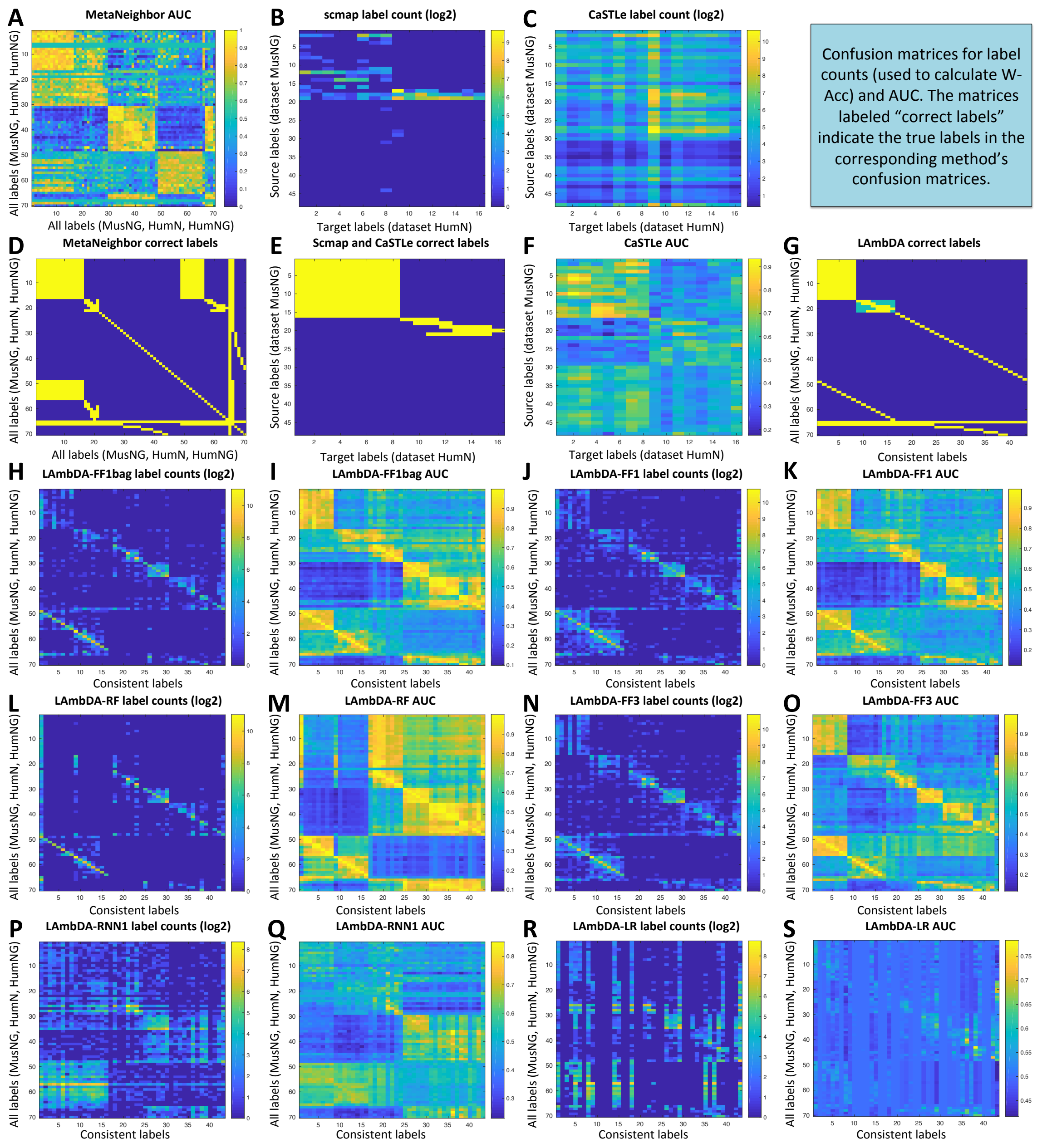
Supplementary Fig. S3. This figure contains all of the confusion matrices for each of the cell classification methods in the two datasets, A (4 labels) and B (3 labels), from simulated 1. A) The AUC confusion matrix for the MetaNeighbor algorithm. Note that all dataset labels are included in the output such that every label is compared against every label. The associated label mask (i.e., correct label mapping) is shown below in D. B) The scmap confusion matrix based on label counts. Note that only one dataset is present in each axis. The associated label mask (i.e., correct label mapping) is shown below in E. C,F) The CaSTLe confusion matrix based on label count and AUC respectively. Note that only one dataset is present on each axis. The associated label mask is shown in E. G) The Label mask used by LAmbDA where the input label mask are all cells in green or yellow. The yellow indicate the true mapping that should be learned. H-S) The LAmbDA confusion matrices for each LAmbDA algorithm type where the label count matrix is on the left and the AUC matrix is on the right. Note that all of the datasets are contained on the y-axis and a smaller set of consistent subtypes are contained on the x-axis. The associated label mask is shown in G.



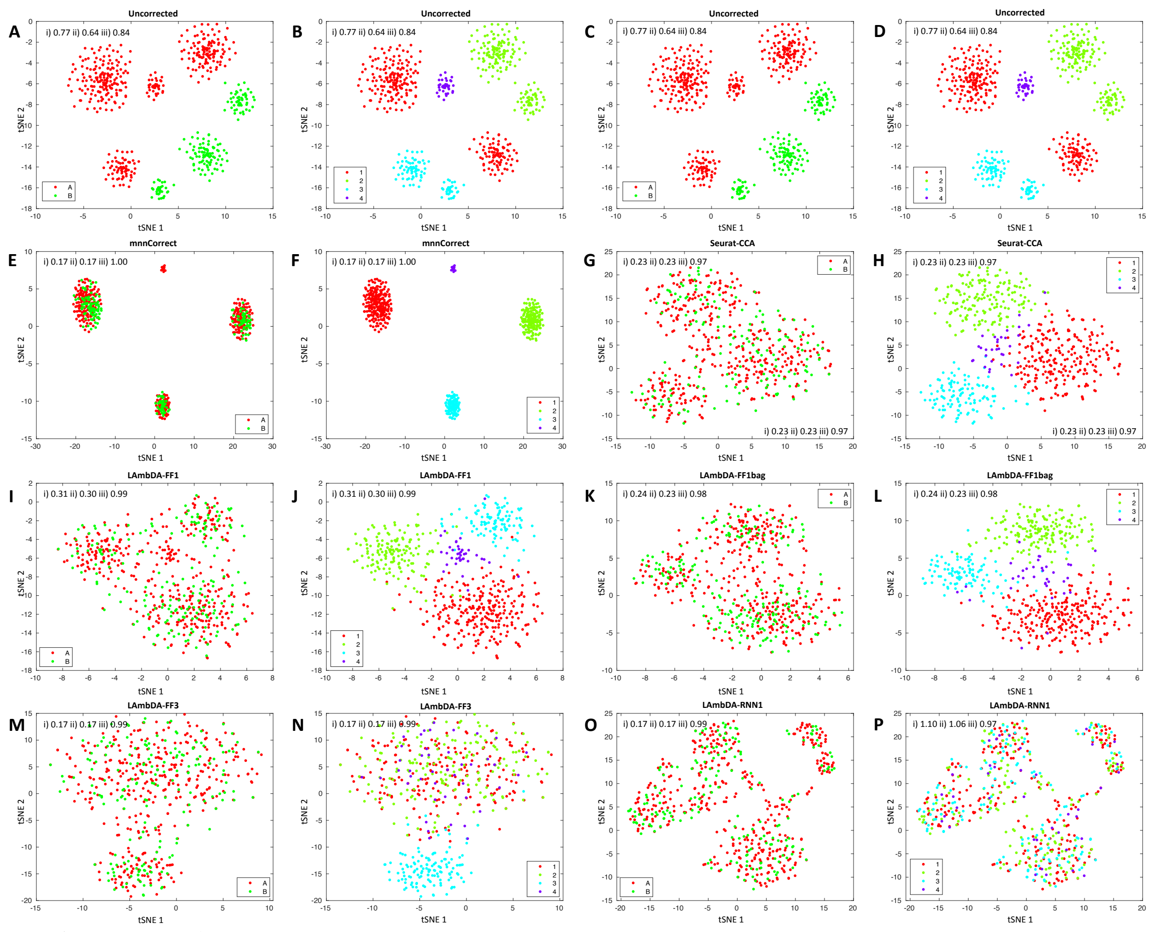
Supplementary Fig. S4. This figure contains all of the confusion matrices for each of the cell classification methods in the two datasets, A (3 labels) and B (3 labels), from simulated 2. A) The AUC confusion matrix for the MetaNeighbor algorithm. Note that all dataset labels are included in the output such that every label is compared against every label. The associated label mask (i.e., correct label mapping) is shown below in D. B) The scmap confusion matrix based on label counts. Note that only one dataset is present in each axis. The associated label mask (i.e., correct label mapping) is shown below in E. C,F) The CaSTLe confusion matrix based on label count and AUC respectively. Note that only one dataset is present on each axis. The associated label mask is shown in E. G) The Label mask used by LAmbDA where the input label mask are all cells in green or yellow. The yellow indicate the true mapping that should be learned. H-S) The LAmbDA confusion matrices for each LAmbDA algorithm type where the label count matrix is on the left and the AUC matrix is on the right. Note that all of the datasets are contained on the y-axis and a smaller set of consistent subtypes are contained on the x-axis. The associated label mask is shown in G.



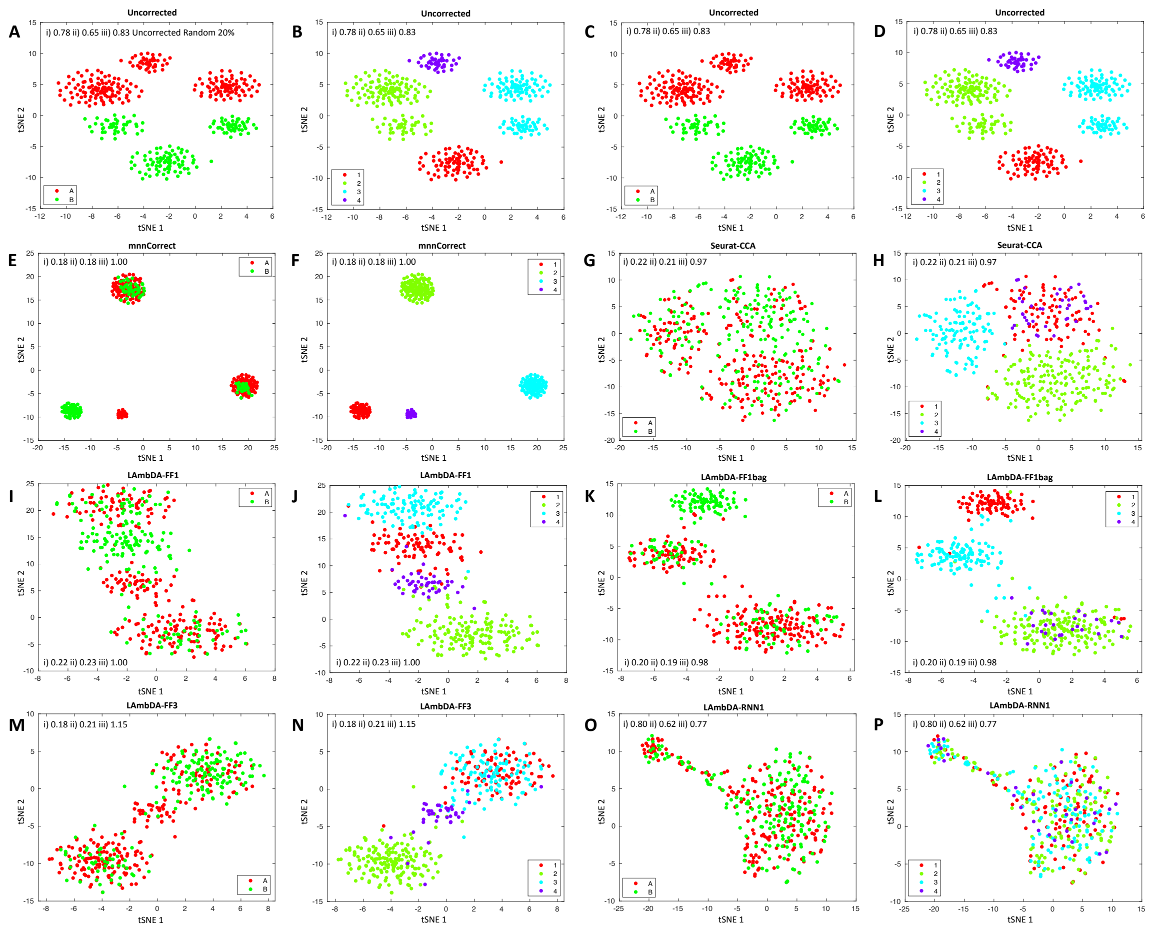
Supplementary Fig. S5. This figure contains all of the confusion matrices for each of the cell classification methods in the three pancreas datasets, Segerstolpe (Seger), Muraro (Mur), and Baron (Bar). A) The AUC confusion matrix for the MetaNeighbor algorithm. Note that all dataset labels are included in the output such that every label is compared against every label. The associated label mask (i.e., correct label mapping) is shown below in D. B) The scmap confusion matrix based on label counts. Note that only one dataset is present in each axis. The associated label mask (i.e., correct label mapping) is shown below in E. C,F) The CaSTLe confusion matrix based on label count and AUC respectively. Note that only one dataset is present on each axis. The associated label mask is shown in E. G) The Label mask used by LAmbDA where the input label mask are all cells in green or yellow. The yellow indicate the true mapping that should be learned. H-S) The LAmbDA confusion matrices for each LAmbDA algorithm type where the label count matrix is on the left and the AUC matrix is on the right. Note that all of the datasets are contained on the y-axis and a smaller set of consistent subtypes are contained on the x-axis. The associated label mask is shown in G.



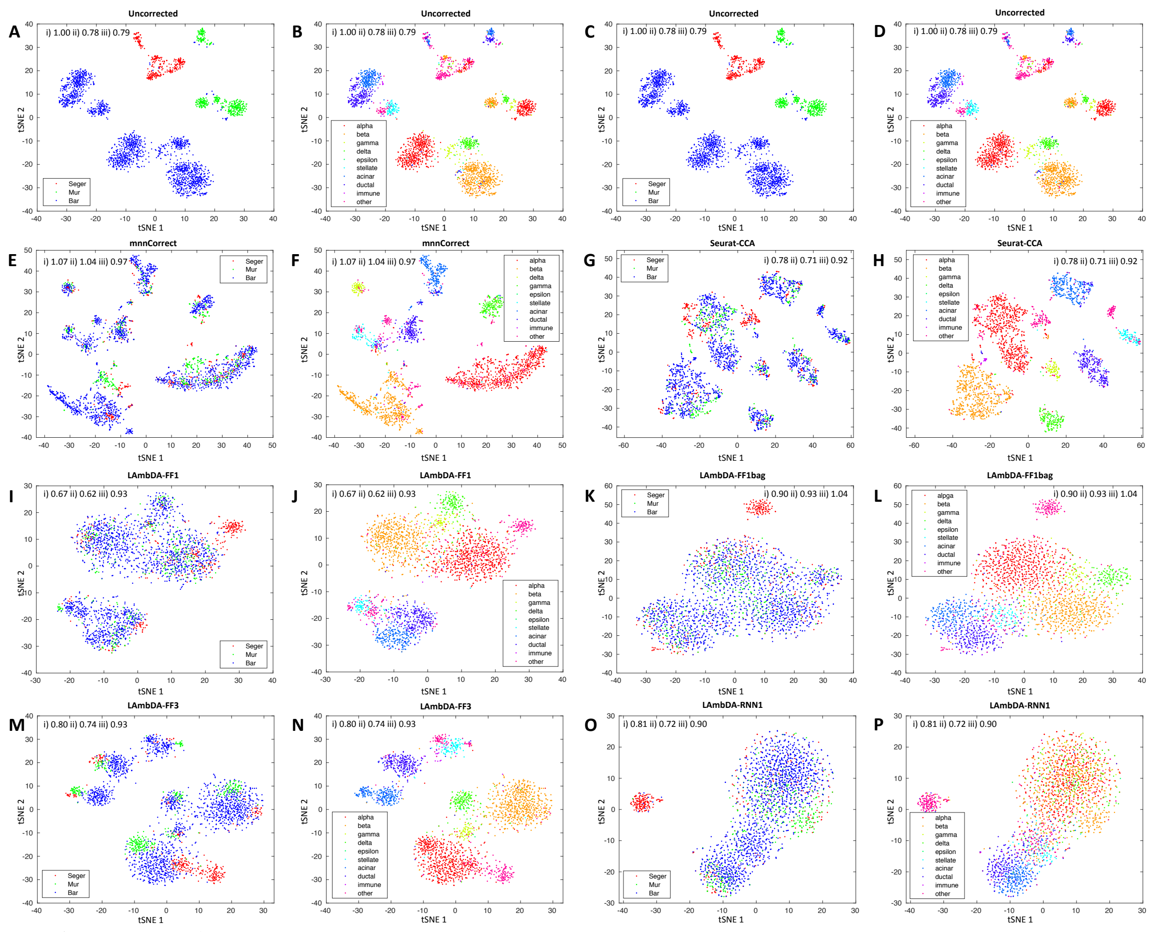
Supplementary Fig. S6. This figure contains all of the confusion matrices for each of the cell classification methods in the three pancreas datasets, Zeisel (MusNG), Lake (HumN), and Darmanis (HumNG). A) The AUC confusion matrix for the MetaNeighbor algorithm. Note that all dataset labels are included in the output such that every label is compared against every label. The associated label mask (i.e., correct label mapping) is shown below in D. B) The scmap confusion matrix based on label counts. Note that only one dataset is present in each axis. The associated label mask (i.e., correct label mapping) is shown below in E. C,F) The CaSTLe confusion matrix based on label count and AUC respectively. Note that only one dataset is present on each axis. The associated label mask is shown in E. G) The Label mask used by LAmbDA where the input label mask are all cells in green or yellow. The yellow indicate the true mapping that should be learned. H-S) The LAmbDA confusion matrices for each LAmbDA algorithm type where the label count matrix is on the left and the AUC matrix is on the right. Note that all of the datasets are contained on the y-axis and a smaller set of consistent subtypes are contained on the x-axis. The associated label mask is shown in G.



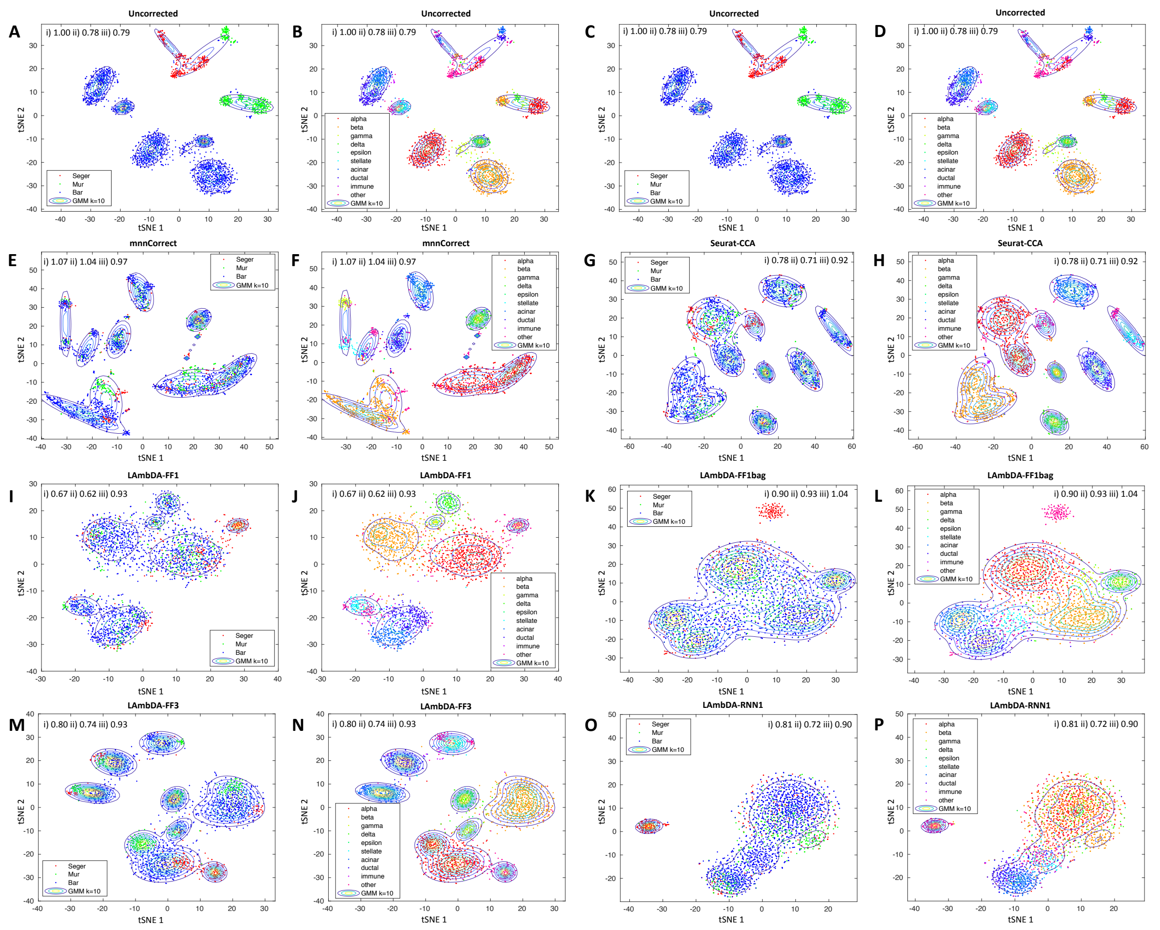
Supplementary Fig. S7. tSNE plots for the simulated 1 datasets (A and B) corrected by each method, Uncorrected, Seurat-CCA, mnnCorrect, and LAmbDA. A,C) These are the uncorrected tSNE plots colored by the dataset (A or B). B,D) These are the uncorrected tSNE plots colored by the cell type (1-4). E,F) The tSNE plots from the output of mnnCorrect colored by dataset in E and cell type in F. G,H) The tSNE plots from the output of Seurat-CCA colored by dataset in G and by cell type in H. I-P) The tSNE plots from the LAmbDA final hidden layer on the validation set of samples colored by dataset in I,K,M,O and by cell type in J,L,N,P. The numbers on the plots specify the distance ratio measure of batch effect correction in each of the representations. Note that for Uncorrected, mnnCorrect, and Seurat-CCA only 20% of the samples were used so that the plots were comparable against the 20% validation set in LAmbDA. The letters i,ii, and iii, indicate the distance metrics for that specific set of cells across all dimensions (not only the tSNE dimensions). The first distance ratio (i), $\boldsymbol{Dat^{-}Sub^{+}/Dat^{+}Sub^{-}}$, represents the ratio of the median different-dataset-same-subtype centroid distance to the median same-dataset-different-subtype centroid distance. The second distance ratio (ii), $\boldsymbol{Dat^{-}Sub^{+}/Dat^{-}Sub^{-}}$, represents the ratio of the median different-dataset-same-subtype centroid distance to the median different-dataset-different-subtype centroid distance. The third distance ratio (iii), $\boldsymbol{Dat^{+}Sub^{-}/Dat^{-}Sub^{-}}$, represents the ratio of the median same-dataset-different-subtype centroid distance to the median different-dataset-different-subtype centroid distance.



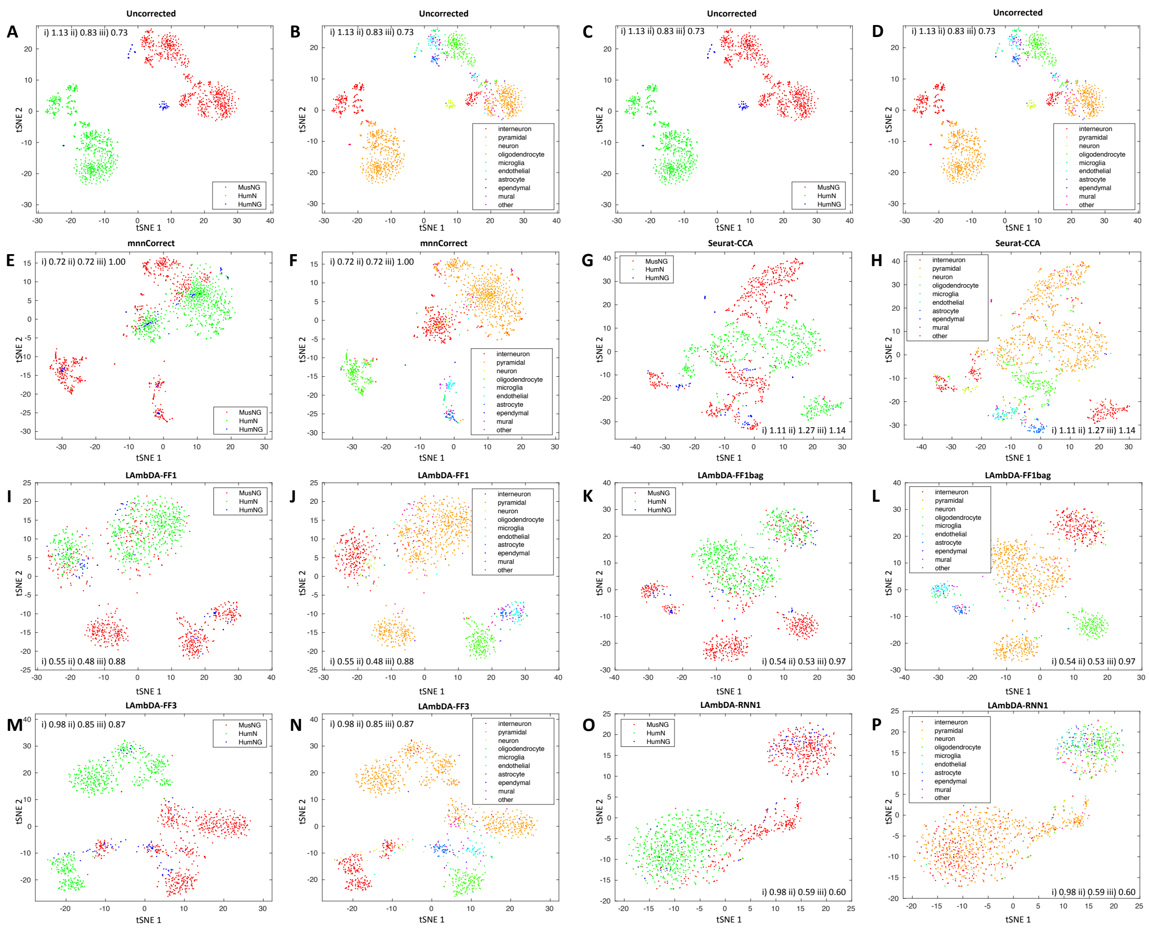
Supplementary Fig. S8. tSNE plots for the simulated 2 datasets (A and B) corrected by each method, Uncorrected, Seurat-CCA, mnnCorrect, and LAmbDA. A,C) These are the uncorrected tSNE plots colored by the dataset (A or B). B,D) These are the uncorrected tSNE plots colored by the cell type (1-4). E,F) The tSNE plots from the output of mnnCorrect colored by dataset in E and cell type in F. G,H) The tSNE plots from the output of Seurat-CCA colored by dataset in G and by cell type in H. I-P) The tSNE plots from the LAmbDA final hidden layer on the validation set of samples colored by dataset in I,K,M,O and by cell type in J,L,N,P. The numbers on the plots specify the distance ratio measure of batch effect correction in each of the representations. Note that for Uncorrected, mnnCorrect, and Seurat-CCA only 20% of the samples were used so that the plots were comparable against the 20% validation set in LAmbDA. The letters i,ii, and iii, indicate the distance metrics for that specific set of cells across all dimensions (not only the tSNE dimensions). The first distance ratio (i), $\boldsymbol{Dat^{-}Sub^{+}/Dat^{+}Sub^{-}}$, represents the ratio of the median different-dataset-same-subtype centroid distance to the median same-dataset-different-subtype centroid distance. The second distance ratio (ii), $\boldsymbol{Dat^{-}Sub^{+}/Dat^{-}Sub^{-}}$, represents the ratio of the median different-dataset-same-subtype centroid distance to the median different-dataset-different-subtype centroid distance. The third distance ratio (iii), $\boldsymbol{Dat^{+}Sub^{-}/Dat^{-}Sub^{-}}$, represents the ratio of the median same-dataset-different-subtype centroid distance to the median different-dataset-different-subtype centroid distance.



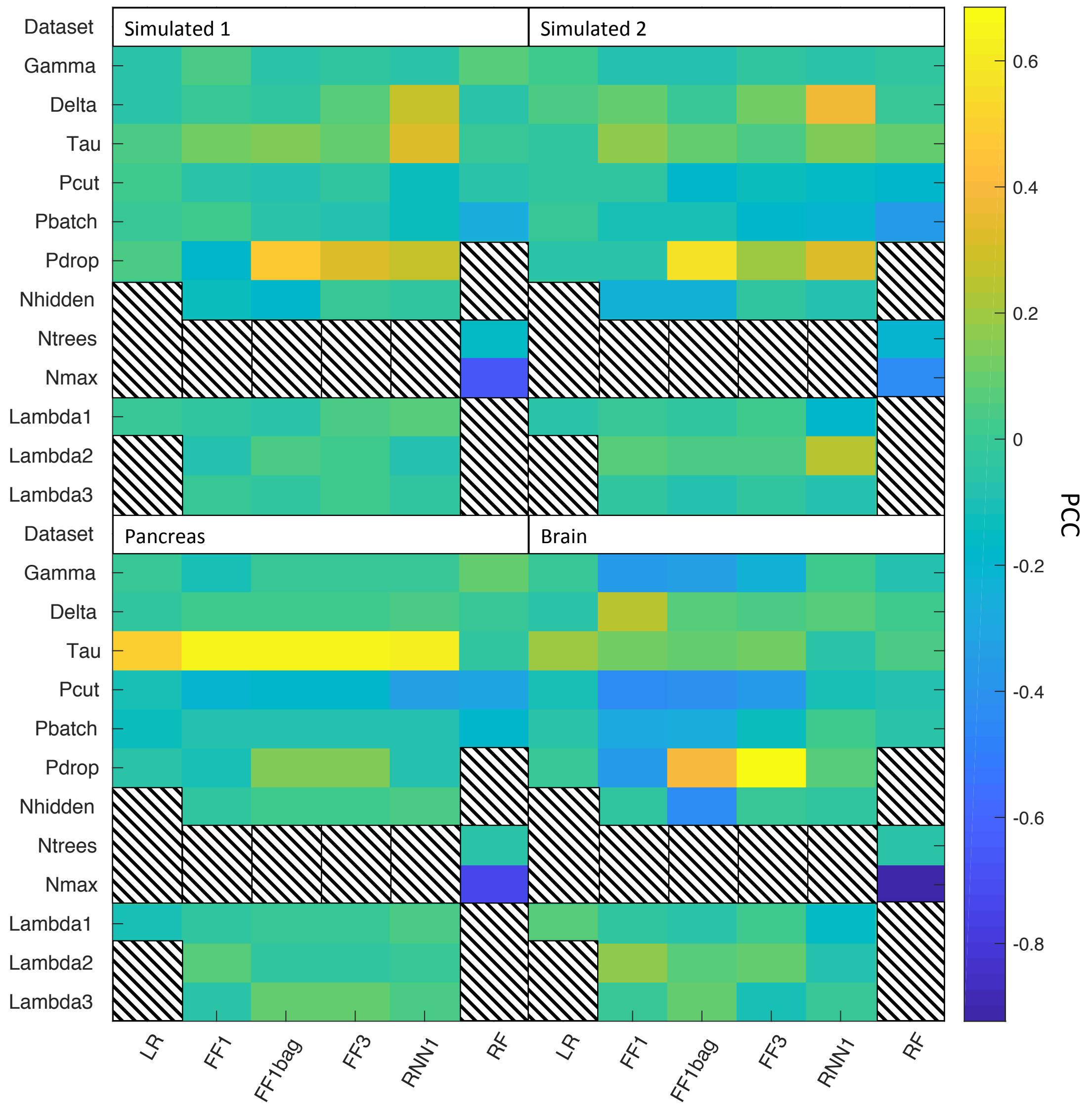
Supplementary Fig. S9. tSNE plots for the pancreas datasets (Segerstolpe (Seger), Muraro (Mur), and Baron (Bar)) corrected by each method, Uncorrected, Seurat-CCA, mnnCorrect, and LAmbDA. A,C) These are the uncorrected tSNE plots colored by the dataset (Seger, Mur, or Bar). B,D) These are the uncorrected tSNE plots colored by the cell type (alpha, beta, gamma, delta, epsilon, stellate, acinar, ductal, immune, and other). E,F) The tSNE plots from the output of mnnCorrect colored by dataset in E and cell type in F. G,H) The tSNE plots from the output of Seurat-CCA colored by dataset in G and by cell type in H. I-P) The tSNE plots from the LAmbDA final hidden layer on the validation set of samples colored by dataset in I,K,M,O and by cell type in J,L,N,P. The numbers on the plots specify the distance ratio measure of batch effect correction in each of the representations. Note that for Uncorrected, mnnCorrect, and Seurat-CCA only 20% of the samples were used so that the plots were comparable against the 20% validation set in LAmbDA. The letters i,ii, and iii, indicate the distance metrics for that specific set of cells across all dimensions (not only the tSNE dimensions). The first distance ratio (i), $\boldsymbol{Dat^{-}Sub^{+}/Dat^{+}Sub^{-}}$, represents the ratio of the median different-dataset-same-subtype centroid distance to the median same-dataset-different-subtype centroid distance. The second distance ratio (ii), $\boldsymbol{Dat^{-}Sub^{+}/Dat^{-}Sub^{-}}$, represents the ratio of the median different-dataset-same-subtype centroid distance to the median different-dataset-different-subtype centroid distance. The third distance ratio (iii), $\boldsymbol{Dat^{+}Sub^{-}/Dat^{-}Sub^{-}}$, represents the ratio of the median same-dataset-different-subtype centroid distance to the median different-dataset-different-subtype centroid distance.



Supplementary Fig. S10. tSNE plots for the pancreas datasets (Segerstolpe (Seger), Muraro (Mur), and Baron (Bar)) corrected by each method, Uncorrected, Seurat-CCA, mnnCorrect, and LAmbDA. Gaussian mixture models were also fit to the tSNE plots to show the stratification of cell types and not dataset as a representative example. A,C) These are the uncorrected tSNE plots colored by the dataset (Seger, Mur, or Bar). B,D) These are the uncorrected tSNE plots colored by the cell type (alpha, beta, gamma, delta, epsilon, stellate, acinar, ductal, immune, and other). E,F) The tSNE plots from the output of mnnCorrect colored by dataset in E and cell type in F. G,H) The tSNE plots from the output of Seurat-CCA colored by dataset in G and by cell type in H. I-P) The tSNE plots from the LAmbDA final hidden layer on the validation set of samples colored by dataset in I,K,M,O and by cell type in J,L,N,P. The numbers on the plots specify the distance ratio measure of batch effect correction in each of the representations. Note that for Uncorrected, mnnCorrect, and Seurat-CCA only 20% of the samples were used so that the plots were comparable against the 20% validation set in LAmbDA. The letters i,ii, and iii, indicate the distance metrics for that specific set of cells across all dimensions (not only the tSNE dimensions). The first distance ratio (i), $\boldsymbol{Dat^{-}Sub^{+}/Dat^{+}Sub^{-}}$, represents the ratio of the median different-dataset-same-subtype centroid distance to the median same-dataset-different-subtype centroid distance. The second distance ratio (ii), $\boldsymbol{Dat^{-}Sub^{+}/Dat^{-}Sub^{-}}$, represents the ratio of the median different-dataset-same-subtype centroid distance to the median different-dataset-different-subtype centroid distance. The third distance ratio (iii), $\boldsymbol{Dat^{+}Sub^{-}/Dat^{-}Sub^{-}}$, represents the ratio of the median same-dataset-different-subtype centroid distance to the median different-dataset-different-subtype centroid distance.



Supplementary Fig. S11. tSNE plots for the brain datasets (Zeisel (MusNG), Lake (HumN), and Darmanis (HumNG)) corrected by each method, Uncorrected, Seurat-CCA, mnnCorrect, and LAmbDA. Gaussian mixture models were also fit to the tSNE plots to show the stratification of cell types and not dataset. A,C) These are the uncorrected tSNE plots colored by the dataset (MusNG, HumN, or HumNG). B,D) These are the uncorrected tSNE plots colored by the cell type (interneuron, pyramidal, neuron, oligodendrocyte, microglia, endothelial, astrocyte, ependymal, mural, and other). E,F) The tSNE plots from the output of mnnCorrect colored by dataset in E and cell type in F. G,H) The tSNE plots from the output of Seurat-CCA colored by dataset in G and by cell type in H. I-P) The tSNE plots from the LAmbDA final hidden layer on the validation set of samples colored by dataset in I,K,M,O and by cell type in J,L,N,P. The numbers on the plots specify the distance ratio measure of batch effect correction in each of the representations. Note that for Uncorrected, mnnCorrect, and Seurat-CCA only 20% of the samples were used so that the plots were comparable against the 20% validation set in LAmbDA. The letters i,ii, and iii, indicate the distance metrics for that specific set of cells across all dimensions (not only the tSNE dimensions). The first distance ratio (i), $\boldsymbol{Dat^{-}Sub^{+}/Dat^{+}Sub^{-}}$, represents the ratio of the median different-dataset-same-subtype centroid distance to the median same-dataset-different-subtype centroid distance. The second distance ratio (ii), $\boldsymbol{Dat^{-}Sub^{+}/Dat^{-}Sub^{-}}$, represents the ratio of the median different-dataset-same-subtype centroid distance to the median different-dataset-different-subtype centroid distance. The third distance ratio (iii), $\boldsymbol{Dat^{+}Sub^{-}/Dat^{-}Sub^{-}}$, represents the ratio of the median same-dataset-different-subtype centroid distance to the median different-dataset-different-subtype centroid distance.



Supplementary Fig. S12. This figure visualizes the Pearson correlation coefficient between each hyperparameter and error during training. These correlations were determined using the 10 rounds of cross validation with 50 rounds for hyperparameter tuning per round of cross validation resulting in 500 data points for each model and dataset. For LAmbDA-RNN1 only 10 hyperparameter tuning rounds were used resulting in 100 data points. Each row is a hyperparameter and each column corresponds to a model and dataset. Boxes with white stripes indicate hyperparameter-model combinations that did not exist such as tree depth (Nmax) for models other than random forest.

Supplementary Tables

Supplementary Table S1. Complete Cross Dataset Mapping: This table contains the significance tests used to determine the accuracies of labels assigned across datasets. Cell Label indicates the cell counts in a confusion matrix across the two datasets used in the experiment. AUC indicates the AUC that was calculated for the same confusion matrix cell based on the label probability output. Note that AUC is based on binary labels so AUC does not give information about the algorithm's ability to correctly select a single label from multiple labels. Alternatively, the Cell Label column measures the ability to select the correct label and no other labels for a cell. The significance tests (Wilcoxon rank-sum) were used to test whether cell-counts/AUCs were higher in the confusion matrix for correct mappings (i.e., dataset A cell type 1 mapped to dataset B cell type 1) opposed to the incorrect mapping (i.e., dataset A cell type 1 mapped to dataset B cell type 2). Italicized values indicate significant test statistics. Bold values indicate the best metric in that particular test across all of the methods. Grey boxes indicate areas that are not available from an algorithm.

Cross-dataset mapping Significance tests		Simulated 1 (7 to 4)		Simulated 2 (6 to 4)		Pancreas (39 to 17)		Brain (70 to 43)	
		Cell label	AUC	Cell label	AUC	Cell label	AUC	Cell label	AUC
LAmbDA-LR	t-test	0.4240	0.4738	0.2458	0.8789	0.9853	*0.0061*	0.9807	0.9686
	Wilcoxon	***<0.0001***	*0.0351*	***0.0002***	**0.0937**	0.9470	*0.0056*	1.0000	0.4007
LAmbDA-RF	t-test	0.6720	*0.0001*	0.4539	0.2973	0.5111	*0.0002*	0.3201	0.2395
	Wilcoxon	0.7611	*0.0351*	0.7128	**0.0937**	0.5150	*0.0015*	0.2386	0.4983
LAmbDA-FF1	t-test	*0.0097*	*0.0008*	0.0620	*0.0062*	*0.0148*	***<0.0001***	*0.0146*	*0.0020*
	Wilcoxon	***<0.0001***	*0.0351*	***0.0002***	**0.0937**	*0.0478*	*0.0001*	***0.0130***	***0.0014***
LAmbDA-FF3	t-test	0.1089	*0.0058*	0.2169	0.1046	*0.0062*	***<0.0001***	0.1982	0.1492
	Wilcoxon	***<0.0001***	*0.0351*	***0.0002***	**0.0937**	*0.0346*	*0.0003*	0.2173	0.1101
LAmbDA-RNN1	t-test	0.3653	0.0735	0.8359	0.9073	0.6277	0.2144	0.3016	0.6701
	Wilcoxon	***<0.0001***	0.0351	***0.0002***	**0.0937**	0.7642	0.2018	0.3305	0.4646
LAmbDA-FF1bag	t-test	***<0.0001***	***<0.0001***	*0.0063*	*0.0047*	*0.0006*	***<0.0001***	***0.0050***	***0.0012***
	Wilcoxon	*0.0006*	*0.0351*	*0.0005*	**0.0937**	*0.0181*	*0.0002*	*0.0175*	*0.0018*
scmap	t-test	***<0.0001***	▨	*0.0103*	▨	***<0.0001***	▨	0.5560	▨
	Wilcoxon	*0.0090*	▨	0.1333	▨	***<0.0001***	▨	0.3671	▨
CaSTLe	t-test	***<0.0001***	*0.0007*	***0.0053***	***0.0043***	*0.0007*	***<0.0001***	0.5628	*0.0019*
	Wilcoxon	*0.0160*	***0.0091***	0.1333	0.1333	*0.0008*	***<0.0001***	0.4688	*0.0049*
MetaNeighbor	t-test	▨	*0.0002*	▨	0.2026	▨	*0.0002*	▨	0.5642
	Wilcoxon	▨	***0.0091***	▨	0.2727	▨	*0.0005*	▨	0.4813

Supplementary Table S2. Mouse and Human Interneuron consistent subtype mapping with overlapping biomarkers from the original publications(Lake, et al., 2016; Zeisel, et al., 2015). The relationships listed in the first two columns can be seen in **Fig. 3G box 1**. In1-8 are the interneuron subtypes from the HumN dataset and Int1-16 are the interneuron subtypes from the MusNG dataset.

HumN	MusNG	Biomarkers
In1	Int6-8,12	CCK,CXCL14
In2	Int5,6,10	CXCL14,VIP
In3	Int9,10,13	CALB2
In4	Int12,14-16	CXCL14,RELN
In5	Int4,14	
In6	Int1-4,13	LHX6,PVALB
In7	Int1,14,15	NPY
In8	Int1,2,4	LHX6,SST,CALB1

References

Aymeric, D. 2018. https://github.com/aymericdamien/TensorFlow-Examples
Claesen, M., *et al.* Hyperparameter tuning in Python using Optunity. In, *Proceedings of the International Workshop on Technical Computing for Machine Learning and Mathematical Engineering*. 2014. p. 3.
Crow, M., *et al.* Characterizing the replicability of cell types defined by single cell RNA-sequencing data using MetaNeighbor. *Nat Commun* 2018;9(1):884.
Darmanis, S., *et al.* A survey of human brain transcriptome diversity at the single cell level. *Proc Natl Acad Sci U S A* 2015;112(23):7285-7290.
Ding, C. and Peng, H. Minimum redundancy feature selection from microarray gene expression data. *J Bioinform Comput Biol* 2005;3(2):185-205.
Kingma, D.P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* 2014.
Lake, B.B., *et al.* Neuronal subtypes and diversity revealed by single-nucleus RNA sequencing of the human brain. *Science* 2016;352(6293):1586-1590.
Rodner, E., *et al.* Fine-grained Recognition in the Noisy Wild: Sensitivity Analysis of Convolutional Neural Networks Approaches. *British Machine Vision Conference* 2016.
Wen, Y., *et al.* A Discriminative Feature Learning Approach for Deep Face Recognition. In. Cham: Springer International Publishing; 2016. p. 499-515.
Zeisel, A., *et al.* Brain structure. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science* 2015;347(6226):1138-1142.