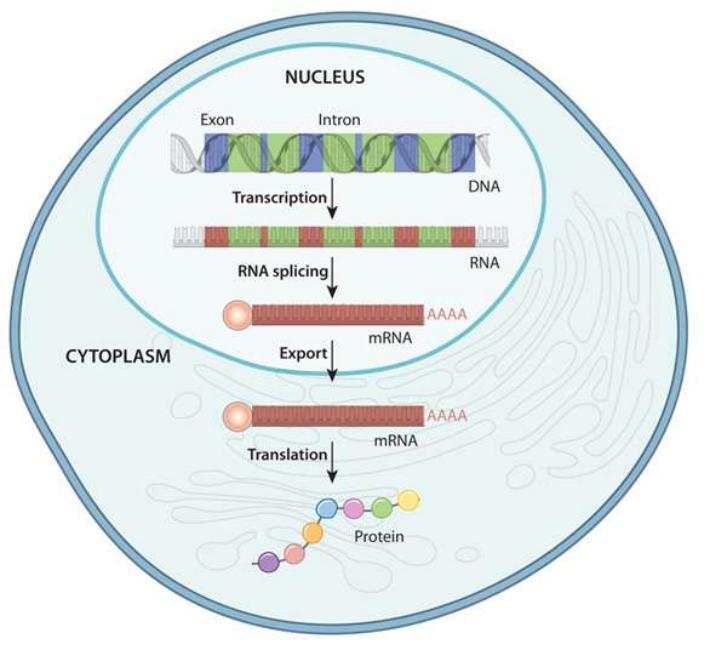
Using neural networks to predict genetic markers of Down Syndrome-associated learning deficits



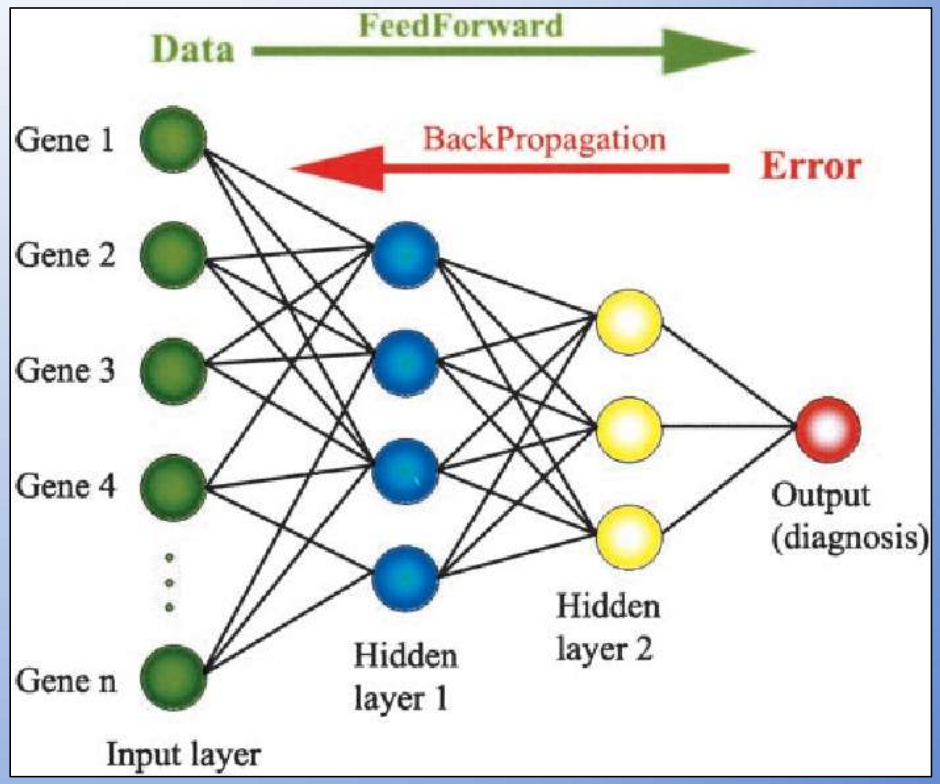
Kevin Potts

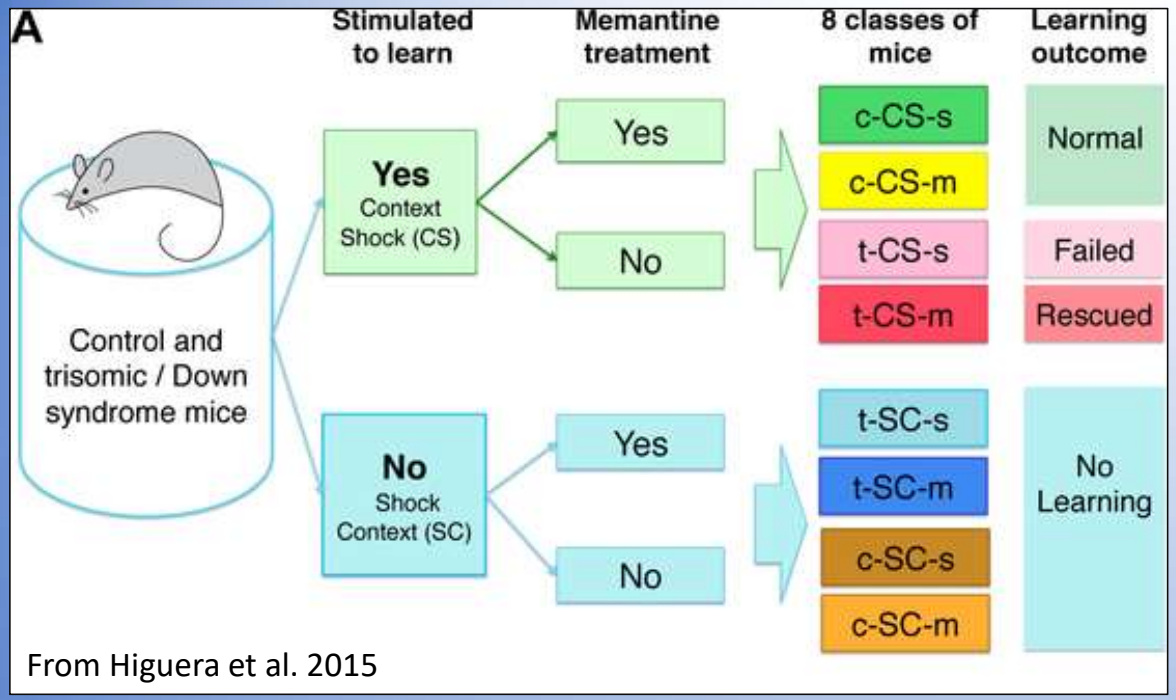
Thinkful Data Science

Final Capstone

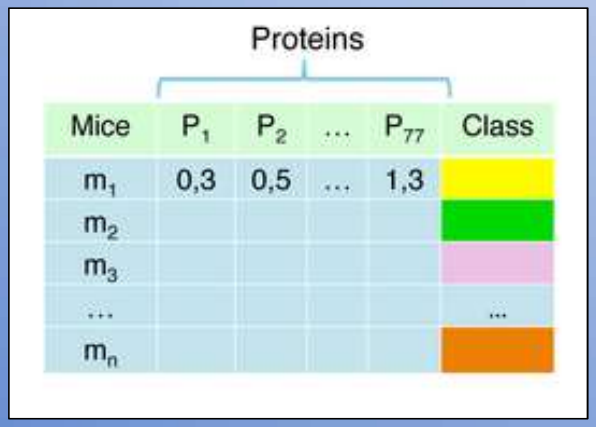
- Down Syndrome (DS) is caused by extra copy of chromosome 21 [Hsa21]
- Neurological problems linked to overexpression of Hsa21 genes
- Pleiotropic effects of Hsa21 genes (the same gene influences multiple traits) – e.g., the same set of genes simultaneously influences learning ability and the expression of DS
  - Complicates the development of targeted treatments for DS-related learning deficits

- Neural networks have recently been employed to detect subtle relationships in polygenic traits (e.g., DS) and states (e.g., the learning process, production of emotion, etc.)
- **Can neural networks be used to simultaneously predict 1) the presence of DS (a polygenic trait) and 2) the process of learning (a polygenic state) based on Hsa1 gene expression values?**
  - A single model that can disentangle pleiotropic relationships could aid in the production of therapies for learning deficits in DS

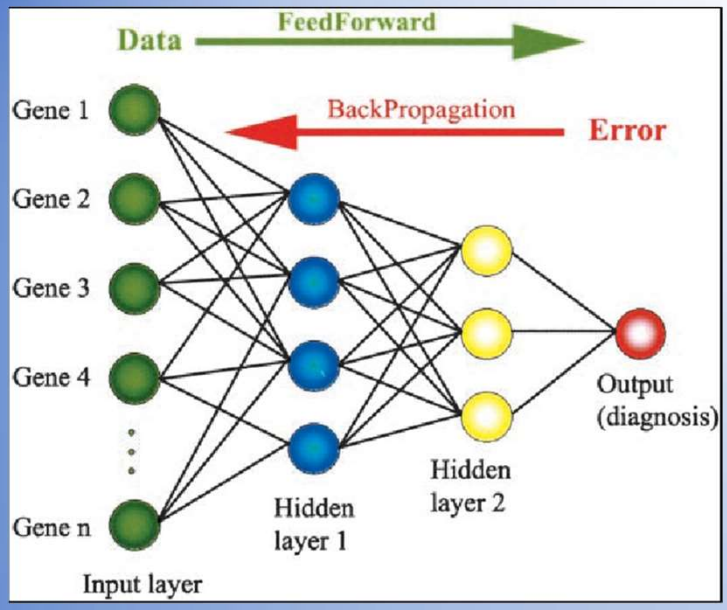


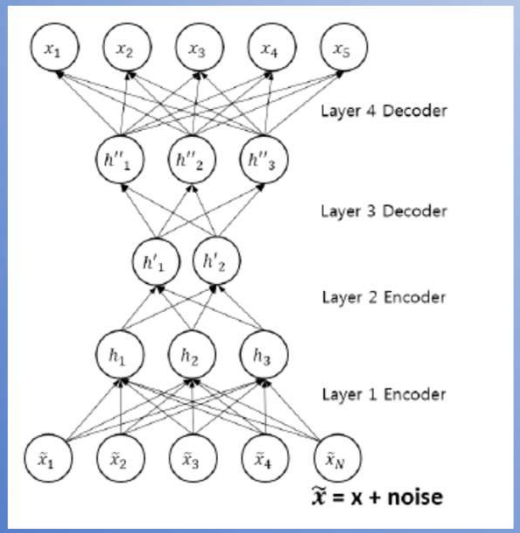


From Higuera et al. 2015



- Data from Higuera et al. 2015 (freely available on UCI Machine Learning Database)
- 1,080 Hsa21 gene expression measurements in lab mice
- 77 features (proteins overexpressed in the brain in Down Syndrome)
- 3 binary experimental variables:
  - — Trisomic/control
  - — CS (context-shock)/SC (shock-context)– whether the mouse was exposed to a situation in which learning should have been possible
  - — Memantine injection/saline injection- Memantine promotes learning (e.g., in Alzheimer's patients)





Methodology:

- Data cleaning and train/test splitting:
  - Imputed missing protein expression values separately for each protein P_i by assigning each missing value a random number drawn from a normal distribution with $\sim N(\mu_{P_i}, \sigma^2_{P_i})$
  - Re-scaled feature values
  - Made a training set by randomly choosing 60% of cases
- Neural network construction
  - Developed two NN architectures: 1) sequential deep NN; 2) denoising autoencoder NN
    - Ran autoencoded X data through a ridge classifier to predict outcome labels
  - Used NN with best general predictive ability to make predictions for specific questions of biological relevance

Sequential neural network

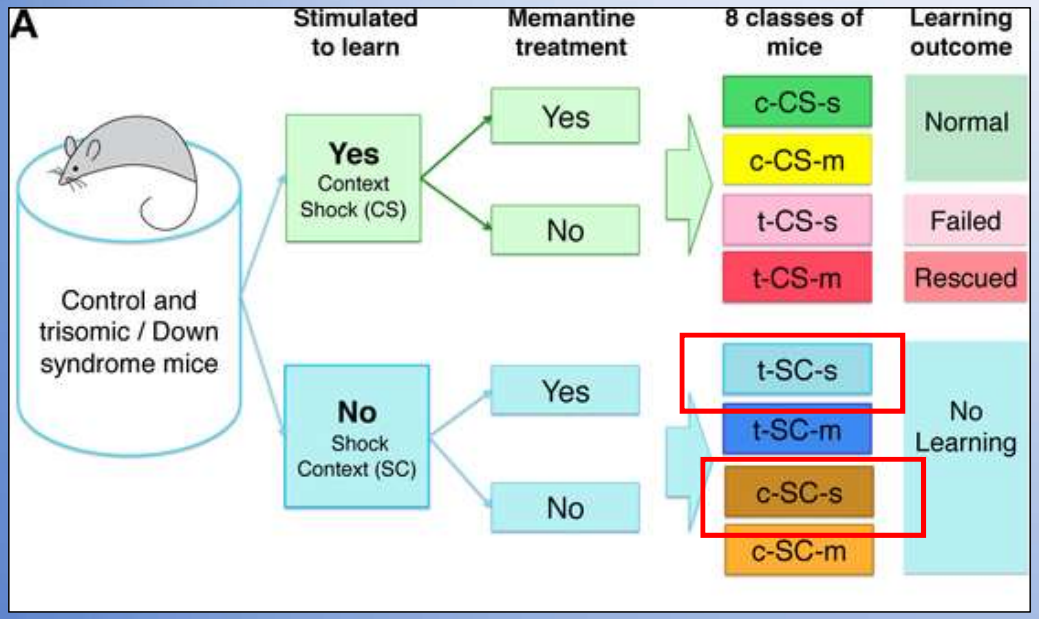
```
def gene_exp_network(X_train, y_train, X_test, y_test):

  models = []

  model = Sequential()
  model.add(Dense(64, activation = 'relu', input_shape = (X_train.shape[1],)))
  model.add(Dropout(0.3))
  model.add(Dense(32, activation = 'relu'))
  model.add(Dropout(0.3))
  model.add(Dense(10, activation = 'relu'))

  model.add(Dense(y_test.shape[1], activation = 'sigmoid'))

  model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

  models.append(model)

  history = model.fit(X_train, y_train,
                      batch_size= 50,
                      epochs=100,
                      verbose=False,
                      validation_data=(X_test, y_test))
  score = model.evaluate(X_test, y_test, verbose=0)
  loss = score[0]
  accuracy = score[1]
  return history, loss, accuracy
```

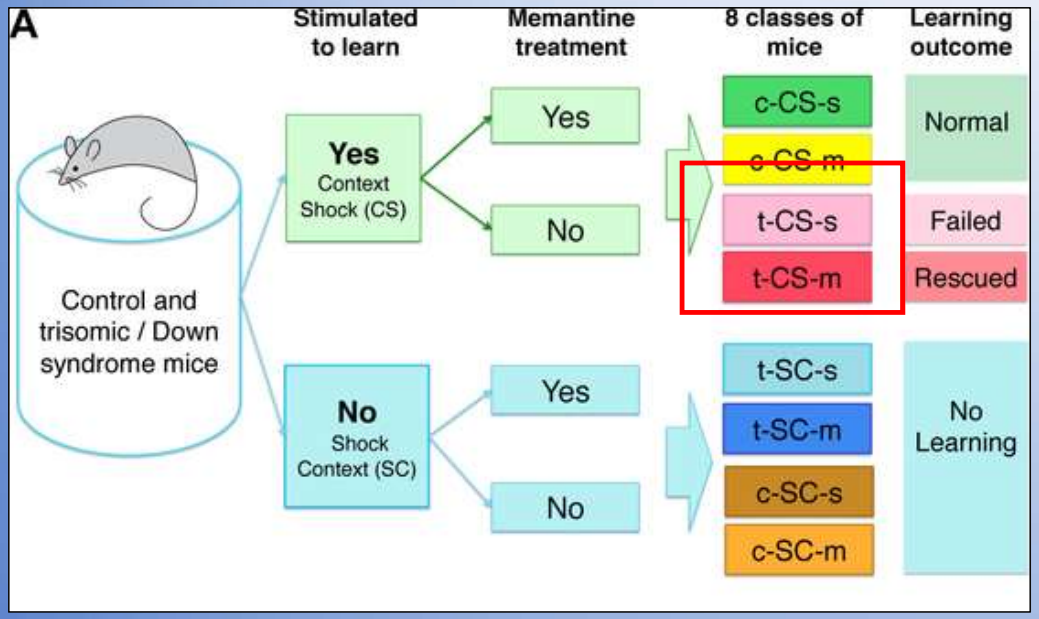
“Denoising” autoencoder

In [0]:

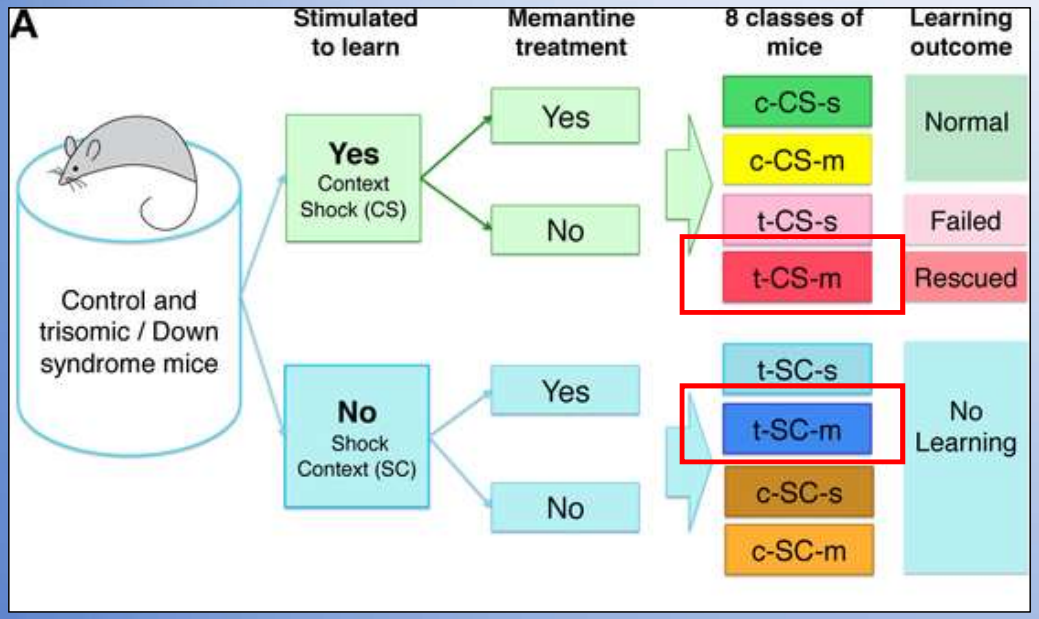
```
#Define an autoencoder function with up to 4 hidden layers

def autoencoder_transform(X_train, X_test, n_hidden):

  input_dim = X_train.shape[1]
  input_layer = Input(shape=(input_dim, ))

  hidden_1 = Dense(input_dim, activation='relu')(input_layer)
  hidden_2 = Dense(input_dim, activation = 'relu')(hidden_1)
  hidden_3 = Dense(input_dim, activation = 'relu')(hidden_2)
  hidden_4 = Dense(input_dim, activation = 'relu')(hidden_3)

  if n_hidden == 1:
    output_layer = output_layer = Dense(input_dim, activation = 'sigmoid')(hidden_1)
  elif n_hidden == 2:
    output_layer = Dense(input_dim, activation = 'sigmoid')(hidden_2)
  elif n_hidden == 3:
    output_layer = Dense(input_dim, activation = 'sigmoid')(hidden_3)
  elif n_hidden == 4:
    output_layer = Dense(input_dim, activation = 'sigmoid')(hidden_4)

  autoencoder = Model(inputs=input_layer, outputs=output_layer)

  autoencoder.compile(optimizer='adam',
                      loss='mean_squared_error',
                      metrics=['accuracy'])


  history = autoencoder.fit(X_train, X_train, validation_data = (X_test, X_test), epochs = 100, verbose =
False).history
  transformed = autoencoder.predict(X_test)

  return history, transformed
```

1) Effect of trisomy, after controlling for other variables?

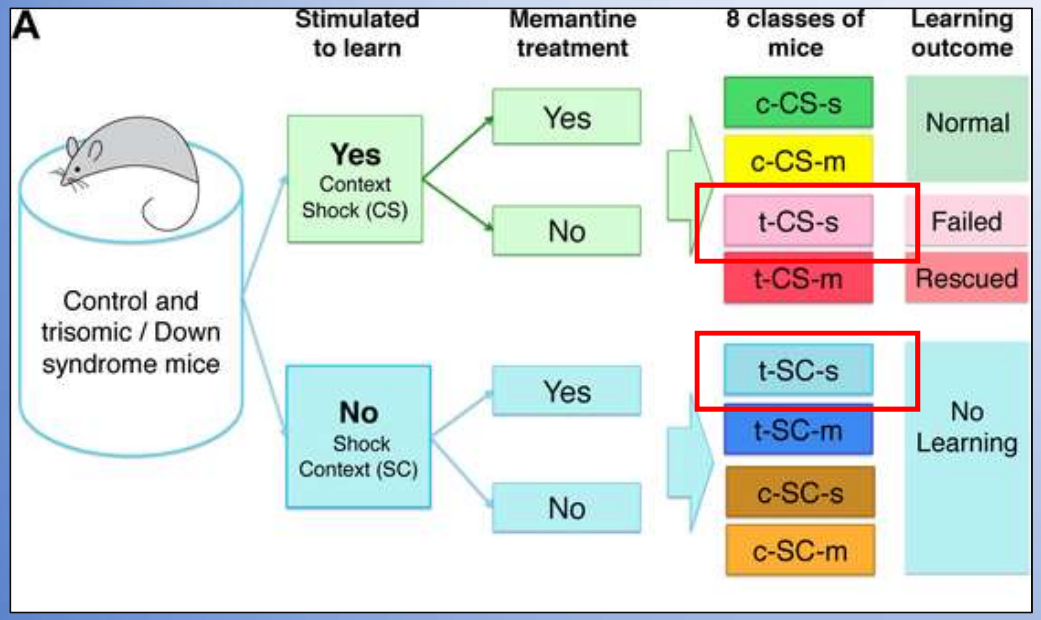




1) Effect of trisomy, after controlling for other variables?
2) Influence of memantine (to rescue learning ability) in trisomic mice



1) Effect of trisomy, after controlling for other variables?
2) Influence of memantine (to rescue learning ability) in trisomic mice
3) Effects of context-fear conditioning (learning) in trisomic mice injected with memantine

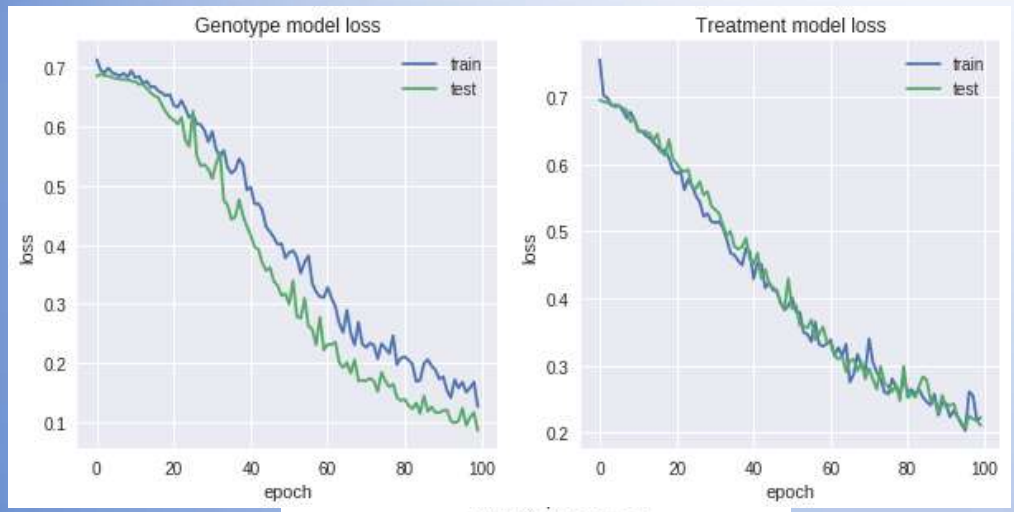


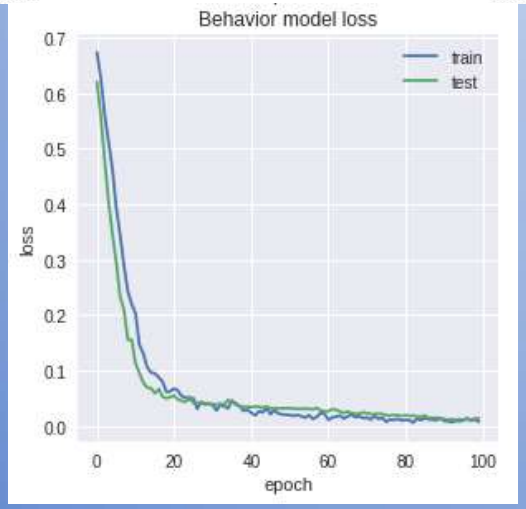
1) Effect of trisomy, after controlling for other variables?
2) Influence of memantine (to rescue learning ability) in trisomic mice
3) Effects of context-fear conditioning (learning) in trisomic mice injected with memantine
4) The influence of context-fear conditioning (learning) on trisomic individuals that have NOT had their learning ability rescued via memantine?

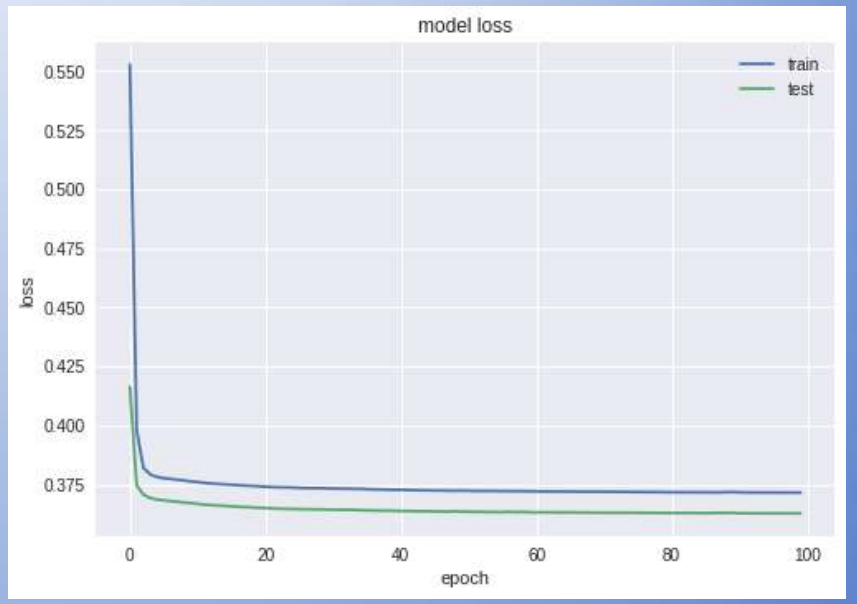
Results

	Autoencoder/ridge accuracy	Sequential NN accuracy
General predictions	Genotype: 0.734 Memantine: 0.690 Learning: 0.993	Genotype: 0.984 Memantine: 0.934 Learning: 0.995

Results – general predictions



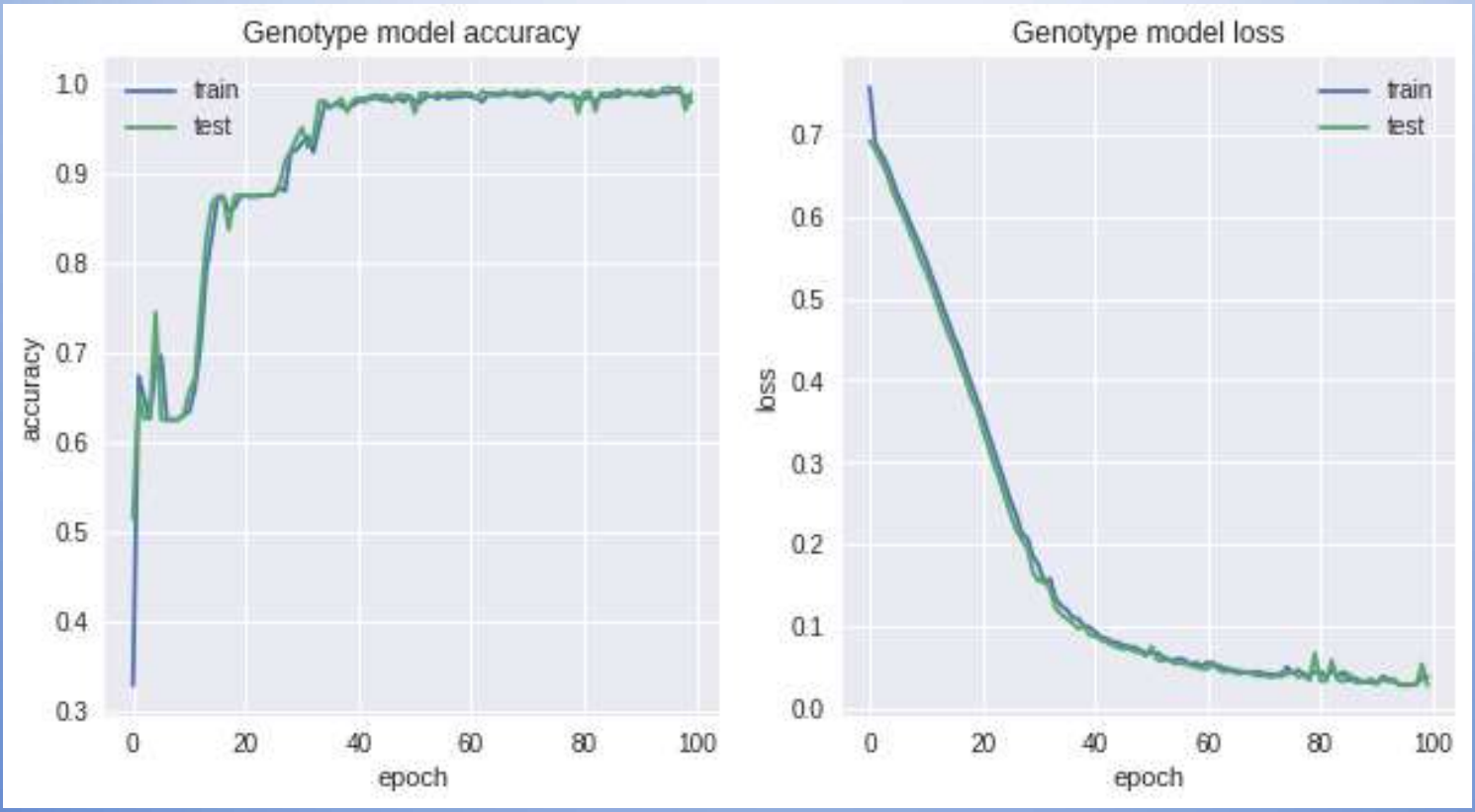




Results

	Autoencoder/ridge accuracy	Sequential NN accuracy
General predictions	Genotype: 0.734 Memantine: 0.690 Learning: 0.993	Genotype: 0.984 Memantine: 0.934 Learning: 0.995
Down Syndrome (controlling for memantine and learning)	--	0.988
	--	
	--	
	--	

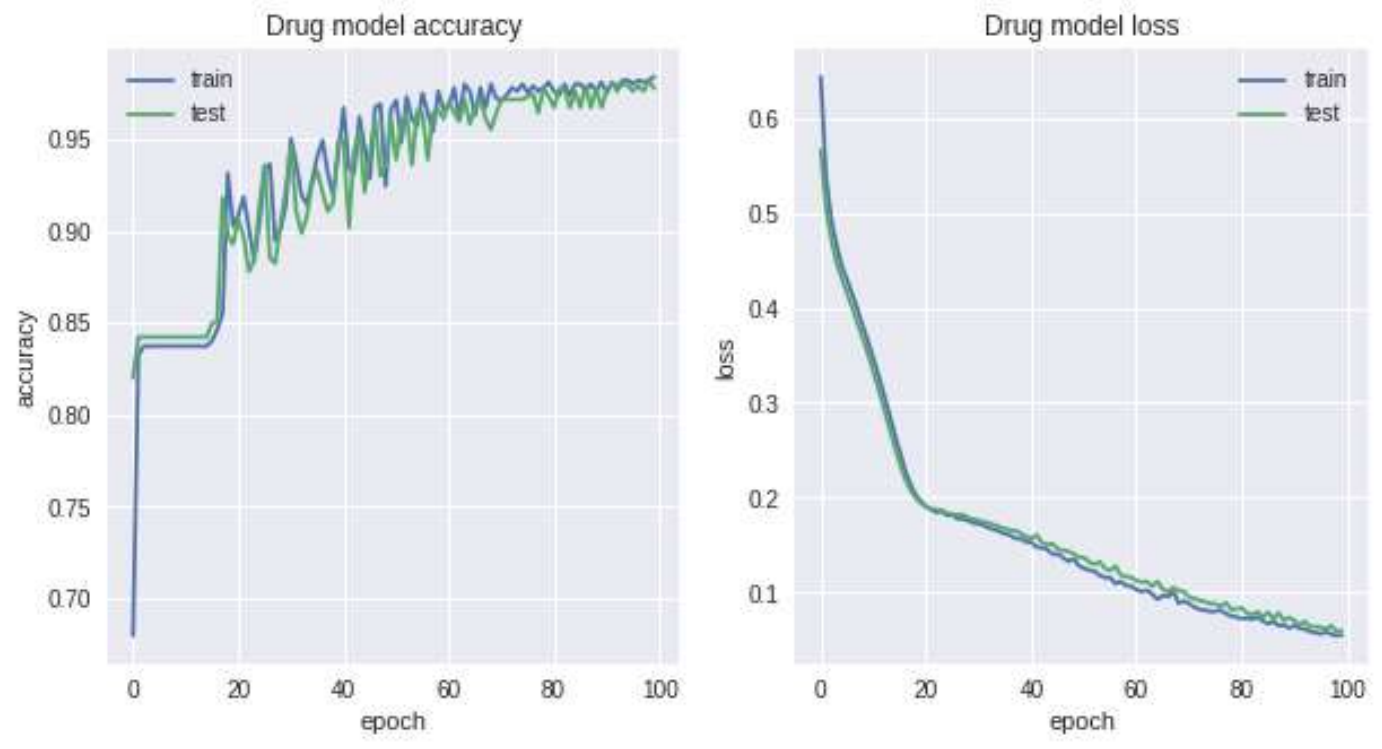
Results – Down Syndrome



Results

	Autoencoder/ridge accuracy	Sequential NN accuracy
General predictions	Genotype: 0.734 Memantine: 0.690 Learning: 0.993	Genotype: 0.984 Memantine: 0.934 Learning: 0.995
Down Syndrome (controlling for memantine and learning)	--	0.988
Effects of memantine on mice with DS	--	0.977
	--	
	--	

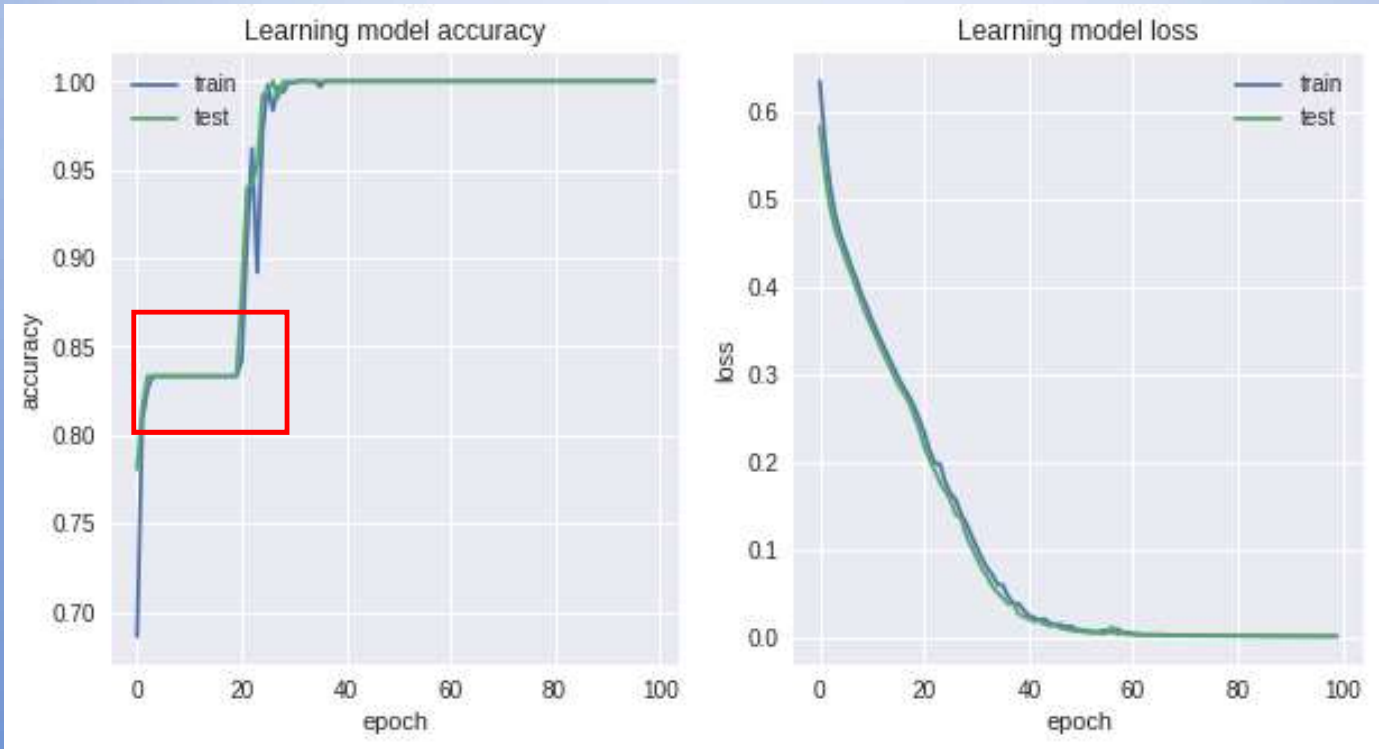
Results – Memantine in mice with DS



Results

	Autoencoder/ridge accuracy	Sequential NN accuracy
General predictions	Genotype: 0.734 Memantine: 0.690 Learning: 0.993	Genotype: 0.984 Memantine: 0.934 Learning: 0.995
Down Syndrome (controlling for memantine and learning)	--	0.988
Effects of memantine on mice with DS	--	0.977
Learning in DS mice treated with memantine	--	1.0 (?)
	--	

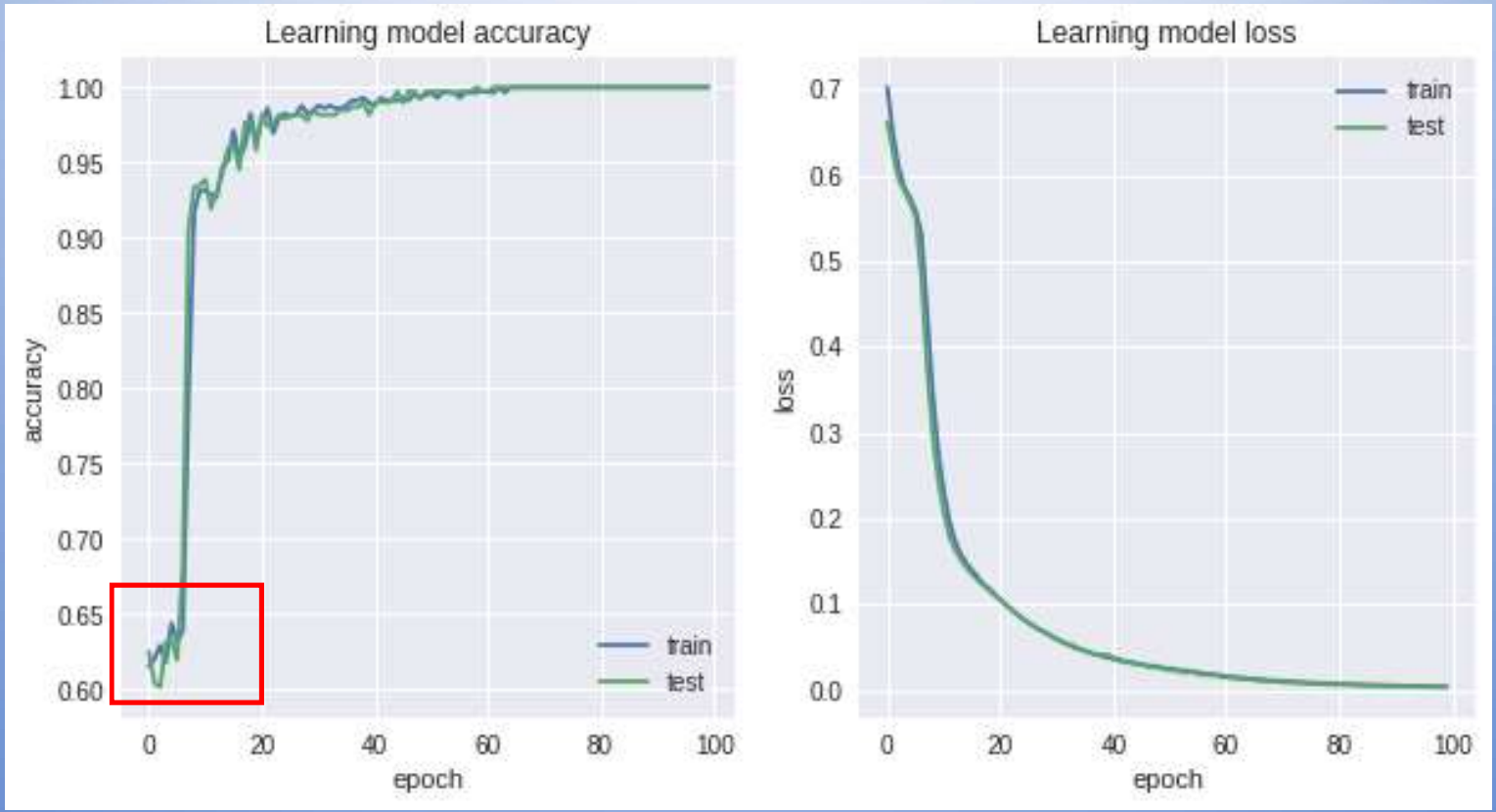
Results – Learning in DS mice treated with memantine



Results

	Autoencoder/ridge accuracy	Sequential NN accuracy
General predictions	Genotype: 0.734 Memantine: 0.690 Learning: 0.993	Genotype: 0.984 Memantine: 0.934 Learning: 0.995
Down Syndrome (controlling for memantine and learning)	--	0.988
Effects of memantine on mice with DS	--	0.977
Learning in DS mice treated with memantine	--	1.0 (?)
Learning in DS mice NOT treated with memantine	--	1.0 (?)

Results – Learning in DS mice NOT treated with memantine



Summary/Discussion

- "Deep" sequential NN outperforms autoencoded X data run through a ridge classifier
  - —"Denoising" removes data that is not actually noise? (synergistic effects among genes, etc)
- Effect of memantine-rescued learning in DS mice is easily detectable in Hsa1 gene expression levels
  - —Starting point for therapies aimed at reducing gene overexpression?
- Effect of exposure to learning is easily predicted, independent of memantine injection status
  - —Artifact of overfitting to the "behavior/learning" outcome?
  - —Cross validation tests: most folds produced scores of ~1.0
  - —Context-fear conditioning has stereotypic effects on gene expression?

THANK YOU!

ANY QUESTIONS?