Run the first 4 cells to import data and create training and testing set

In [2]:

```
import numpy as np
import h5py
import tensorflow as tf
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import SimpleRNN,LSTM, Dense, Activation
from keras.utils import to_categorical
#from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from UtilNNDL import *
```

In [3]:

```
"""
#file_path = '/home/carla/Downloads/project_datasets/project_datasets/'
file_path = '/home/kunal/Desktop/FinalProject/datasets/A01T_slice.mat'

A01T = h5py.File(file_path,'r')
data = np.copy(A01T['image'])
data = np.transpose(data,(0,2,1))
data = data[:,:,:22]
labels = np.copy(A01T['type'])
labels = labels[0,0:data.shape[0]:1]
labels = np.asarray(labels, dtype=np.int32)

a = data[:56]
b = data[57:]
data = np.vstack((a,b))
a = labels[:56]
b = labels[57:]
labels = np.hstack((a,b))
#enc = OneHotEncoder()
#enc_labels = enc.fit_transform(labels.reshape(-1,1)).toarray()
enc_labels = to_categorical(labels-769, num_classes=4)
print(enc_labels)

#scaler = StandardScaler()
#data = scaler.fit_transform(data,enc_labels)
"""
```

Out[3]:

```
"\n#file_path = '/home/carla/Downloads/project_datasets/project_datasets/'\
nfile_path = '/home/kunal/Desktop/FinalProject/datasets/A01T_slice.mat'\n\n
A01T = h5py.File(file_path,'r')\ndata = np.copy(A01T['image'])\ndata = np.t
ranspose(data,(0,2,1))\ndata = data[:,:,:22]\nlabels = np.copy(A01T['type']
)\nlabels = labels[0,0:data.shape[0]:1]\nlabels = np.asarray(labels, dtype=
np.int32)\n\na = data[:56]\nb = data[57:]\ndata = np.vstack((a,b))\na = lab
els[:56]\nb = labels[57:]\nlabels = np.hstack((a,b))\n#enc = OneHotEncoder(
)\n#enc_labels = enc.fit_transform(labels.reshape(-1,1)).toarray()\nenc_lab
els = to_categorical(labels-769, num_classes=4)\nprint(enc_labels)\n\n#scal
er = StandardScaler()\n#data = scaler.fit_transform(data,enc_labels)\n"
```

In [4]:

```
"""
bs, t, f = data.shape
np.random.seed(42)
shuffle = np.random.choice(bs,bs,replace=False)

train_samples = 237
train_data = data[shuffle[:train_samples],:,:]
train_labels = enc_labels[shuffle[:train_samples]]
test_data = data[shuffle[train_samples:],:,:]
test_labels =enc_labels[shuffle[train_samples:]]

train_data = np.transpose(train_data,(0,2,1))
test_data = np.transpose(test_data,(0,2,1))

train_data,train_labels = create_window_data(train_data,train_labels)
test_data,test_labels = create_window_data(test_data,test_labels)

train_data = np.transpose(train_data,(0,2,1))
test_data = np.transpose(test_data,(0,2,1))

bs, t, f = train_data.shape
"""
```

Out[4]:

```
'\nbs, t, f = data.shape\nnp.random.seed(42)\nshuffle = np.random.choice(bs
,bs,replace=False)\n\ntrain_samples = 237\ntrain_data = data[shuffle[:train
_samples],:,:]\ntrain_labels = enc_labels[shuffle[:train_samples]]\ntest_da
ta = data[shuffle[train_samples:],:,:]\ntest_labels =enc_labels[shuffle[tra
in_samples:]]\n\ntrain_data = np.transpose(train_data,(0,2,1))\ntest_data =
np.transpose(test_data,(0,2,1))\n\ntrain_data,train_labels = create_window_
data(train_data,train_labels)\ntest_data,test_labels = create_window_data(t
est_data,test_labels)\n\ntrain_data = np.transpose(train_data,(0,2,1))\ntes
t_data = np.transpose(test_data,(0,2,1))\n\nbs, t, f = train_data.shape\n'
```

In [6]:

```
#Prepare the data by taking out nans and dividing into test and train
file_path = '/home/carla/Downloads/project_datasets/project_datasets/'
#file_path = '/home/kunal/Desktop/FinalProject/datasets/'
train_data, test_data, train_labels, test_labels = prepare_data(file_path,
                                                                num_test_sa
mples = 50,
                                                                verbose= Fa
lse,
                                                                return_all=
True,
                                                                num_files =
1)
print train_data.shape
print train_labels.shape
print test_data.shape
print test_labels.shape
```

```
(237, 22, 1000)
(237, 4)
(50, 22, 1000)
(50, 4)
```

In [7]:

```
#assist numerical stability
train_data = train_data*(1e6)
test_data = test_data*(1e6)
```

In [8]:

```
#Bandpass filter the data
train_data = train_data.swapaxes(1,2)
test_data = test_data.swapaxes(1,2)
print train_data.shape
print test_data.shape
for i,a in enumerate(train_data):
    train_data[i] = bandpass_cnt(a, 4, 38, 250, filt_order=3)
for i,a in enumerate(test_data):
    test_data[i] = bandpass_cnt(a, 4, 38, 250, filt_order=3)
print train_data.shape
print test_data.shape
```

```
(237, 1000, 22)
(50, 1000, 22)
(237, 1000, 22)
(50, 1000, 22)
```

In [9]:

```
#Standardize the data
for i,a in enumerate(train_data):
    train_data[i] = exponential_running_standardize(a, factor_new=0.001, in
it_block_size=1000, eps=1e-4)
for i,a in enumerate(test_data):
    test_data[i] = exponential_running_standardize(a, factor_new=0.001, ini
t_block_size=1000, eps=1e-4)
train_data = train_data.swapaxes(1,2)
test_data = test_data.swapaxes(1,2)
print train_data.shape
print test_data.shape
```

```
(237, 22, 1000)
(50, 22, 1000)
```

In [10]:

```
#Augment the data into a bigger set by windowing
train_data_sliced, train_labels_sliced = create_window_data(train_data, tra
in_labels, windows=10,window_size=512)
test_data_sliced, test_labels_sliced = create_window_data(test_data, test_l
abels, windows=10)

train_data_sliced = train_data_sliced.swapaxes(1,2)
test_data_sliced = test_data_sliced.swapaxes(1,2)

bs,t,f = train_data_sliced.shape

print train_data_sliced.shape
print train_labels_sliced.shape
print test_data_sliced.shape
print test_labels_sliced.shape
```

```
(2370, 512, 22)
(2370, 4)
(500, 512, 22)
(500, 4)
```

Everything from this point down is Testing

In [18]:

```
model = Sequential([
    LSTM(100, input_shape=(t,f)),
    Dense(32),
    Activation('relu'),
    #Dense(64),
    #Activation('relu'),
    Dense(32),
    Activation('relu'),
    Dense(4),
    Activation('softmax'),
])

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

hist = model.fit(train_data,train_labels,epochs=15,validation_split=0.25,ba
tch_size=32,verbose=0)
test_score = model.evaluate(test_data, test_labels, batch_size=32)

print(test_score)

plot_hist([hist.history['acc'],hist.history['val_acc']],['Training Accuracy
','Val Accuracy'],title='Accuracies')
plot_hist([hist.history['loss'],hist.history['val_loss']],['Training Loss',
'Val Loss'],title='Losses')
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-18-4798cfa1d3bb> in <module>()
     15              metrics=['accuracy'])
     16 
---> 17 hist = model.fit(train_data,train_labels,epochs=15,validation_split
=0.25,batch_size=32,verbose=0)
     18 test_score = model.evaluate(test_data, test_labels, batch_size=32)
     19 

/home/carla/Documents/tensorflow/local/lib/python2.7/site-packages/keras/mo
dels.pyc in fit(self, x, y, batch_size, epochs, verbose, callbacks, validat
ion_split, validation_data, shuffle, class_weight, sample_weight, initial_e
poch, steps_per_epoch, validation_steps, **kwargs)
    961                               initial_epoch=initial_epoch,
    962                               steps_per_epoch=steps_per_epoch,
--> 963                               validation_steps=validation_steps)
    964 
    965     def evaluate(self, x=None, y=None,

/home/carla/Documents/tensorflow/local/lib/python2.7/site-packages/keras/en
gine/training.pyc in fit(self, x, y, batch_size, epochs, verbose, callbacks
, validation_split, validation_data, shuffle, class_weight, sample_weight,
initial_epoch, steps_per_epoch, validation_steps, **kwargs)
   1710                               initial_epoch=initial_epoch,
   1711                               steps_per_epoch=steps_per_epoch,
-> 1712                               validation_steps=validation_steps)
   1713 
   1714     def evaluate(self, x=None, y=None,

/home/carla/Documents/tensorflow/local/lib/python2.7/site-packages/keras/en
gine/training.pyc in _fit_loop(self, f, ins, out_labels, batch_size, epochs
, verbose, callbacks, val_f, val_ins, shuffle, callback_metrics, initial_ep
och, steps_per_epoch, validation_steps)
   1233                         ins_batch[i] = ins_batch[i].toarray()
   1234 
-> 1235                     outs = f(ins_batch)
   1236                     if not isinstance(outs, list):
   1237                         outs = [outs]

/home/carla/Documents/tensorflow/local/lib/python2.7/site-packages/keras/ba
ckend/tensorflow_backend.pyc in __call__(self, inputs)
   2473         session = get_session()
   2474         updated = session.run(fetches=fetches, feed_dict=feed_dict,
-> 2475                               **self.session_kwargs)
   2476         return updated[:len(self.outputs)]
   2477 

/home/carla/Documents/tensorflow/local/lib/python2.7/site-packages/tensorfl
ow/python/client/session.pyc in run(self, fetches, feed_dict, options, run_
metadata)
    893     try:
    894       result = self._run(None, fetches, feed_dict, options_ptr,
--> 895                          run_metadata_ptr)
    896       if run_metadata:
    897         proto_data = tf_session.TF_GetBuffer(run_metadata_ptr)

/home/carla/Documents/tensorflow/local/lib/python2.7/site-packages/tensorfl
ow/python/client/session.pyc in _run(self, handle, fetches, feed_dict, opti
ons, run_metadata)
   1126     if final_fetches or final_targets or (handle and feed_dict_tens
or):
   1127       results = self._do_run(handle, final_targets, final_fetches,
```

Modified VGGnet for this type of data

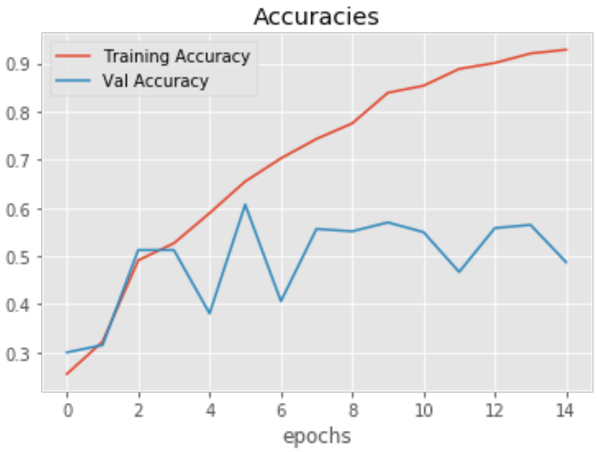
Modified VGG net to handle our input i.e. replace 2D with 1D, etc.(need to check dimensions and might need to transpose input to original shape)

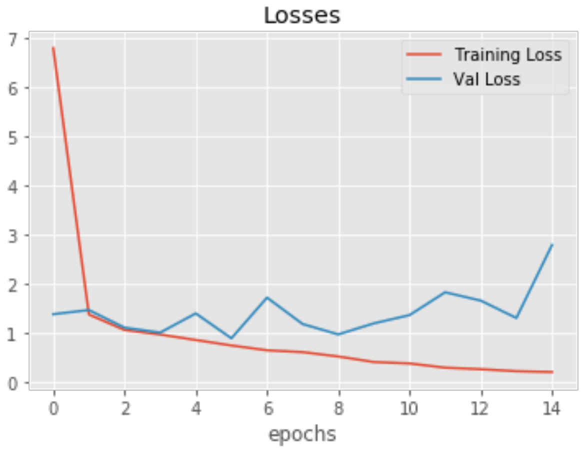
Original VGGnet implementation can be found at hte address below

In [20]:

```
### VGGnet
# https://keras.io/getting-started/sequential-model-guide/#examples

import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv1D, MaxPooling1D, BatchNormalization
from keras.optimizers import SGD

norm_train = np.transpose((-np.mean(train_data,axis=2)+np.transpose(train_d
ata,(2,0,1)))/np.std(train_data,axis=2),(1,2,0))
norm_test = np.transpose((-np.mean(test_data,axis=2)+np.transpose(test_data
,(2,0,1)))/np.std(test_data,axis=2),(1,2,0))

model = Sequential()
#model.add(LSTM(100, input_shape=(t,f)))
model.add(Conv1D(32, 4, activation='relu',input_shape=(t,f)))             #
Originally 32 each
model.add(BatchNormalization())
model.add(Conv1D(32, 4, activation='relu'))
model.add(MaxPooling1D())
model.add(Dropout(0.25))

model.add(Conv1D(64, 4, activation='relu'))
#Originally 64 each
model.add(BatchNormalization())
model.add(Conv1D(64, 4, activation='relu'))
model.add(MaxPooling1D())
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4, activation='softmax'))

#sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
#model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['ac
curacy'])

model.compile(optimizer = 'rmsprop',
             loss = 'categorical_crossentropy',
             metrics=['accuracy'])

hist = model.fit(train_data_sliced,train_labels_sliced,epochs=15,validation
_split=0.25,batch_size=32,verbose=0)
test_score = model.evaluate(test_data_sliced, test_labels_sliced, batch_siz
e=32)

print(test_score)
plot_hist([hist.history['acc'],hist.history['val_acc']],['Training Accuracy
','Val Accuracy'],title='Accuracies')
plot_hist([hist.history['loss'],hist.history['val_loss']],['Training Loss',
'Val Loss'],title='Losses')
```

```
500/500 [==============================] - 0s 514us/step
[2.7438375377655029, 0.47399999999999998]
```

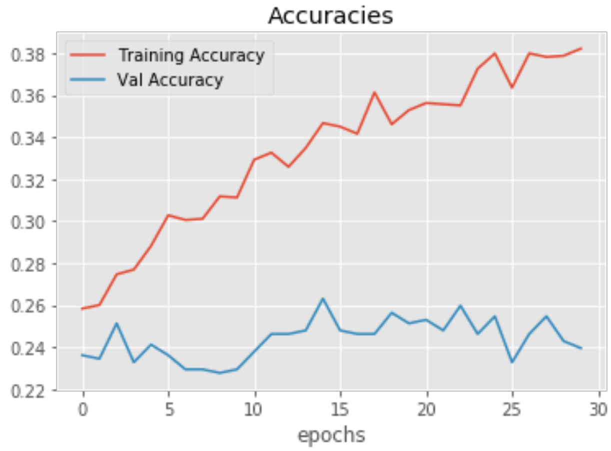


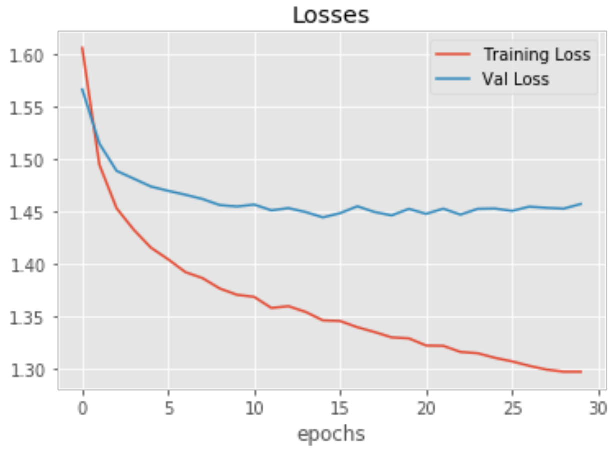


Simple RNN model

```
In [42]: model = Sequential([
             SimpleRNN(64, input_shape=(t,f)),
             Dense(32),
             BatchNormalization(),
             Activation('relu'),
             Dense(4),
             Activation('softmax'),
         ])

         model.compile(optimizer = 'sgd',
                       loss = 'categorical_crossentropy',
                       metrics=['accuracy'])

         hist = model.fit(train_data,train_labels,epochs=30,validation_split=0.25,ba
         tch_size=64,verbose=0)
         test_score = model.evaluate(test_data, test_labels, batch_size=32)

         print(test_score)
         plot_hist([hist.history['acc'],hist.history['val_acc']],['Training Accuracy
         ','Val Accuracy'],title='Accuracies')
         plot_hist([hist.history['loss'],hist.history['val_loss']],['Training Loss',
         'Val Loss'],title='Losses')
```

```
500/500 [==============================] - 1s 1ms/step
[1.438882619857788, 0.25000000005960465]
```





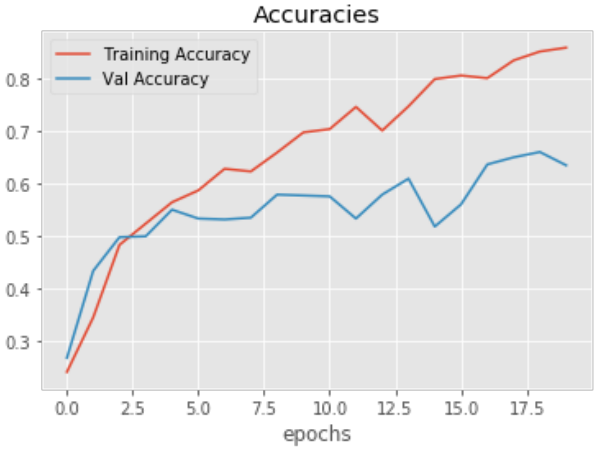
CHRONONET PAPER

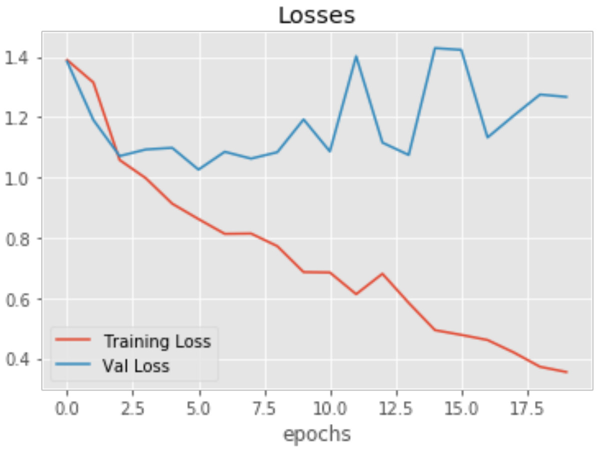
C-RNN implementation (Figure 1b)

In [22]:

```
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv1D, MaxPooling1D, BatchNormalization,GRU
from keras.optimizers import SGD


model = Sequential()

model.add(Conv1D(32, 4, strides=2,activation='relu',input_shape=(t,f)))
model.add(Conv1D(32, 4, strides=2,activation='relu'))
model.add(Conv1D(32, 4, strides=2,activation='relu'))

#model.add(Flatten())

model.add(GRU(32,activation='tanh',return_sequences=True))
model.add(GRU(32,activation='tanh',return_sequences=True))
model.add(GRU(32,activation='tanh',return_sequences=True))
model.add(GRU(32,activation='tanh'))


model.add(Dense(4, activation='softmax'))

#sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
#model.compile(loss='categorical_crossentropy', optimizer=sgd)

#model.add()

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

hist = model.fit(train_data_sliced,train_labels_sliced,epochs=20,validation
_split=0.25,batch_size=32,verbose=0)
test_score = model.evaluate(test_data_sliced,test_labels_sliced, batch_size
=32)
print(test_score)
plot_hist([hist.history['acc'],hist.history['val_acc']],['Training Accuracy
','Val Accuracy'],title='Accuracies')
plot_hist([hist.history['loss'],hist.history['val_loss']],['Training Loss',
'Val Loss'],title='Losses')
```

```
500/500 [==============================] - 0s 939us/step
[1.1306497268676758, 0.6200000009536743 6]
```



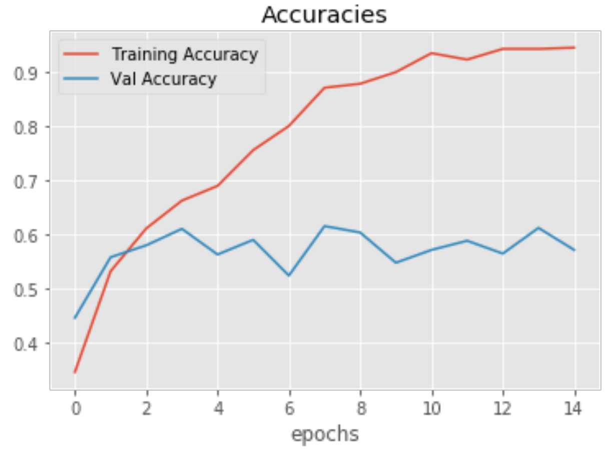


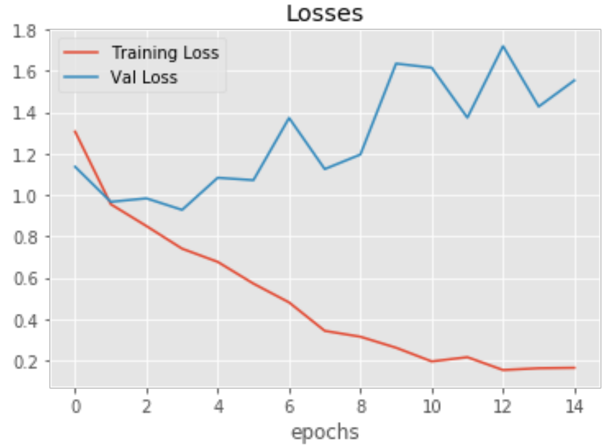
Implementation of Figure 1b but adding regularization structures like that found in VGGnet

In [23]:

```
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv1D, MaxPooling1D, BatchNormalization,GRU
from keras.optimizers import SGD

#norm_train = np.transpose((-np.mean(train_data,axis=2)+np.transpose(train_
data,(2,0,1)))/np.std(train_data,axis=2),(1,2,0))
model = Sequential()

model.add(Conv1D(32, 4, strides=2,activation='relu',input_shape=(t,f)))
model.add(BatchNormalization())                            #From VGGnet
model.add(Conv1D(32, 4, strides=2,activation='relu'))
model.add(BatchNormalization())                            #From VGGnet
model.add(Conv1D(32, 4, strides=2,activation='relu'))
model.add(MaxPooling1D())                                  #From VGGnet
model.add(Dropout(0.25))                                   #From VGGnet
#model.add(Flatten())

model.add(GRU(32,activation='tanh',return_sequences=True))
model.add(GRU(32,activation='tanh',return_sequences=True))
model.add(GRU(32,activation='tanh',return_sequences=True))  #removed becaus
e of overfitting problem to small sample size
model.add(GRU(32,activation='tanh'))

#model.add(Dense(256, activation='relu'))                   #From VGGnet, b
ut makes model suck
#model.add(Dropout(0.5))                                    #From VGGnet, b
ut makes model suck
model.add(Dense(4, activation='softmax'))

# From VGGnet, works well for some reason
#sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
#model.compile(loss='categorical_crossentropy', optimizer=sgd,metrics=['acc
uracy'])

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

#hist.history is a dictionary with all accs and losses
hist = model.fit(train_data_sliced,train_labels_sliced,epochs=15,validation
_split=0.25,batch_size=32,verbose=0)
test_score = model.evaluate(test_data_sliced, test_labels_sliced, batch_siz
e=32)
print(test_score)

plot_hist([hist.history['acc'],hist.history['val_acc']],['Training Accuracy
','Val Accuracy'],title='Accuracies')
plot_hist([hist.history['loss'],hist.history['val_loss']],['Training Loss',
'Val Loss'],title='Losses')
```

```
500/500 [==============================] - 0s 619us/step
[1.6481628928184509, 0.56200000095367431]
```



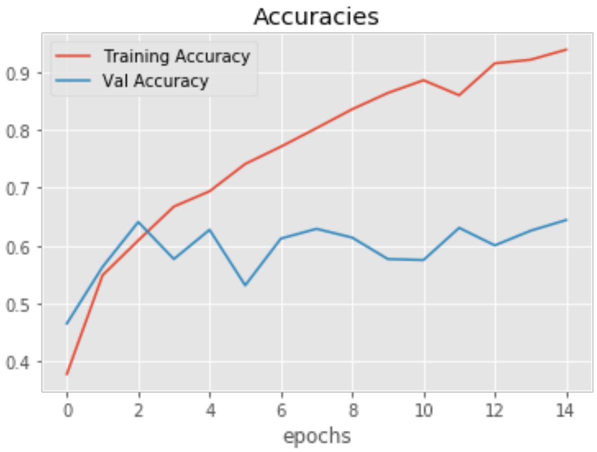


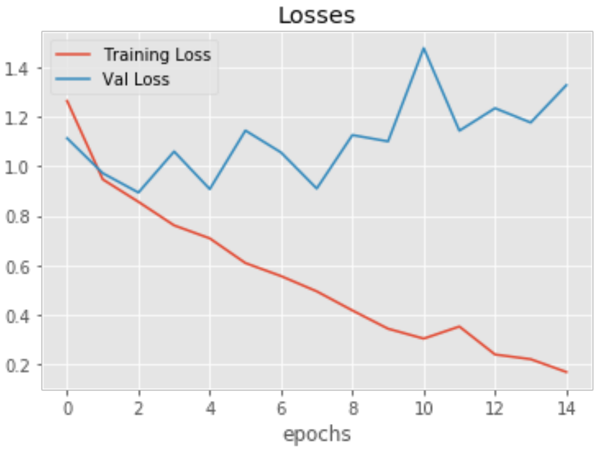
Replaced GRU with LSTM

In [37]:

```
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv1D, MaxPooling1D, BatchNormalization,GRU,LSTM
from keras.optimizers import SGD

#norm_train = np.transpose((-np.mean(train_data,axis=2)+np.transpose(train_
data,(2,0,1)))/np.std(train_data,axis=2),(1,2,0))
model = Sequential()

model.add(Conv1D(32, 4, strides=2,activation='relu',input_shape=(t,f)))
model.add(BatchNormalization())                            #From VGGnet
model.add(Conv1D(32, 4, strides=2,activation='relu'))
model.add(BatchNormalization())                            #From VGGnet
model.add(Conv1D(32, 4, strides=2,activation='relu'))
model.add(MaxPooling1D())                                  #From VGGnet
model.add(Dropout(0.25))                                   #From VGGnet
#model.add(Flatten())

model.add(LSTM(32,activation='tanh',return_sequences=True))
model.add(LSTM(32,activation='tanh',return_sequences=True))
model.add(LSTM(32,activation='tanh',return_sequences=True))
model.add(LSTM(32,activation='tanh'))

#model.add(Dense(256, activation='relu'))                   #From VGGnet, b
ut makes model suck
#model.add(Dropout(0.5))                                    #From VGGnet, b
ut makes model suck
model.add(Dense(4, activation='softmax'))

# From VGGnet, works well for some reason
#sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
#model.compile(loss='categorical_crossentropy', optimizer=sgd,metrics=['acc
uracy'])

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

#hist.history is a dictionary with all accs and losses
hist = model.fit(train_data_sliced,train_labels_sliced,epochs=15,validation
_split=0.25,batch_size=32,verbose=0)
test_score = model.evaluate(test_data_sliced, test_labels_sliced, batch_siz
e=32)

print "Test Results are ", test_score
plot_hist([hist.history['acc'],hist.history['val_acc']],['Training Accuracy
','Val Accuracy'],title='Accuracies')
plot_hist([hist.history['loss'],hist.history['val_loss']],['Training Loss',
'Val Loss'],title='Losses')
```

```
500/500 [==============================] - 0s 667us/step
Test Results are  [0.90564616107940676, 0.71599999999999997]
```



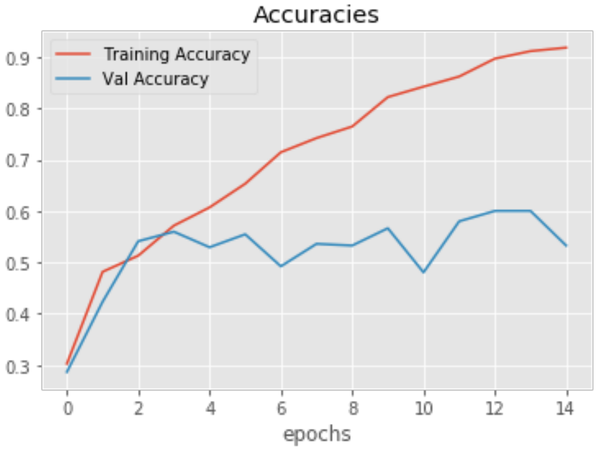


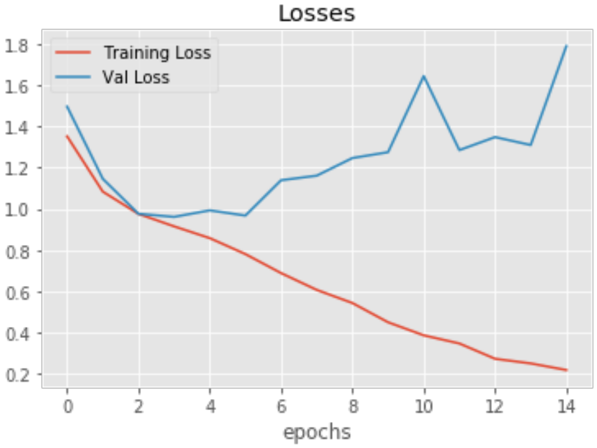
IC-RNN

In [34]:

```
from keras.layers import Input,Dense,concatenate,Flatten,GRU,Conv1D
from keras.models import Model

inputs= Input(shape=(t,f))

# First Inception
tower1 = Conv1D(32, 2, strides=2,activation='relu',padding="causal")(inputs
)
tower2 = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(inputs
)
tower3 = Conv1D(32, 8, strides=2,activation='relu',padding="causal")(inputs
)
x = concatenate([tower1,tower2,tower3],axis=2)

# Second Inception
tower1 = Conv1D(32, 2, strides=2,activation='relu',padding="causal")(x)
tower2 = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(x)
tower3 = Conv1D(32, 8, strides=2,activation='relu',padding="causal")(x)
x = concatenate([tower1,tower2,tower3],axis=2)

# Third Inception
tower1 = Conv1D(32, 2, strides=2,activation='relu',padding="causal")(x)
tower2 = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(x)
tower3 = Conv1D(32, 8, strides=2,activation='relu',padding="causal")(x)
x = concatenate([tower1,tower2,tower3],axis=2)


x = GRU(32,activation='tanh',return_sequences=True)(x)
x = GRU(32,activation='tanh',return_sequences=True)(x)
x = GRU(32,activation='tanh',return_sequences=True)(x)
x = GRU(32,activation='tanh')(x)


predictions = Dense(4,activation='softmax')(x)

model = Model(inputs=inputs, outputs=predictions)

model.compile(optimizer = 'rmsprop',
             loss = 'categorical_crossentropy',
             metrics=['accuracy'])

#hist.history is a dictionary with all accs and losses
hist = model.fit(train_data_sliced,train_labels_sliced,epochs=15,validation
_split=0.25,batch_size=32,verbose=0)
test_score = model.evaluate(test_data_sliced, test_labels_sliced, batch_siz
e=32)

print "Test Results are ", test_score
plot_hist([hist.history['acc'],hist.history['val_acc']],['Training Accuracy
','Val Accuracy'],title='Accuracies')
plot_hist([hist.history['loss'],hist.history['val_loss']],['Training Loss',
'Val Loss'],title='Losses')
```

```
500/500 [==============================] - 1s 1ms/step
Test Results are  [1.7981220669746398, 0.52800000095367428]
```



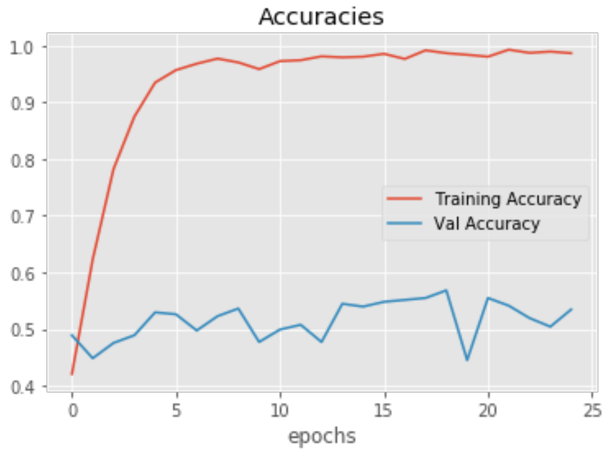


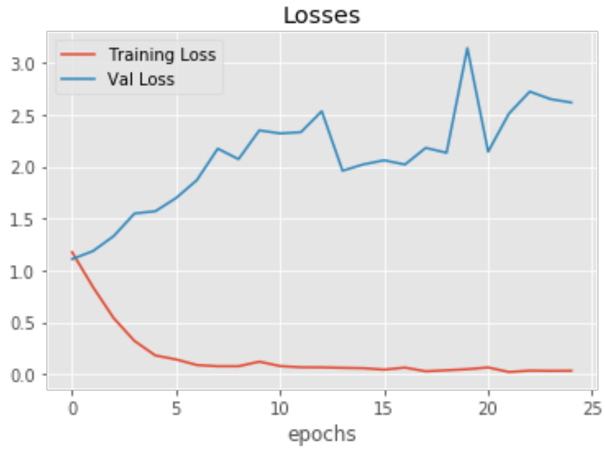
IC-RNN Testing

In [38]:

```
from keras.layers import Input,Dense,concatenate,Flatten,GRU,Conv1D,Bidirectional
from keras.models import Model

inputs= Input(shape=(t,f))

# First Inception
tower1 = Conv1D(32, 2, strides=2,activation='relu',padding="causal")(inputs)
tower1 = BatchNormalization()(tower1)
tower2 = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(inputs)
tower2 = BatchNormalization()(tower2)
tower3 = Conv1D(32, 8, strides=2,activation='relu',padding="causal")(inputs)
tower3 = BatchNormalization()(tower3)
#tower4 = MaxPooling1D()(inputs)
x = concatenate([tower1,tower2,tower3],axis=2)
#x = Dropout(0.5)(x)

# Second Inception
tower1 = Conv1D(32, 2, strides=2,activation='relu',padding="causal")(x)
tower1 = BatchNormalization()(tower1)
tower2 = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(x)
tower2 = BatchNormalization()(tower2)
tower3 = Conv1D(32, 8, strides=2,activation='relu',padding="causal")(x)
tower3 = BatchNormalization()(tower3)
#tower4 = MaxPooling1D()(x)
x = concatenate([tower1,tower2,tower3],axis=2)
#x = Dropout(0.5)(x)


# Third Inception
tower1 = Conv1D(32, 2, strides=2,activation='relu',padding="causal")(x)
tower1 = BatchNormalization()(tower1)
tower2 = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(x)
tower2 = BatchNormalization()(tower2)
tower3 = Conv1D(32, 8, strides=2,activation='relu',padding="causal")(x)
tower3 = BatchNormalization()(tower3)
#tower4 = MaxPooling1D()(x)
x = concatenate([tower1,tower2,tower3],axis=2)
#x = Dropout(0.5)(x)


x = (GRU(32,activation='tanh',return_sequences=True))(x)
x = (GRU(32,activation='tanh',return_sequences=True))(x)
x = (GRU(32,activation='tanh',return_sequences=True))(x)
x = (GRU(32,activation='tanh'))(x)


predictions = Dense(4,activation='softmax')(x)

model = Model(inputs=inputs, outputs=predictions)

model.compile(optimizer = 'rmsprop',
             loss = 'categorical_crossentropy',
             metrics=['accuracy'])

#hist.history is a dictionary with all accs and losses
hist = model.fit(train_data_sliced,train_labels_sliced,epochs=25,validation_split=0.25,batch_size=32,verbose=0)
test_score = model.evaluate(test_data_sliced, test_labels_sliced, batch_size=32)
```

```
500/500 [==============================] - 1s 1ms/step
Test Results are  [2.7042381000518798, 0.5060000047683713]
```



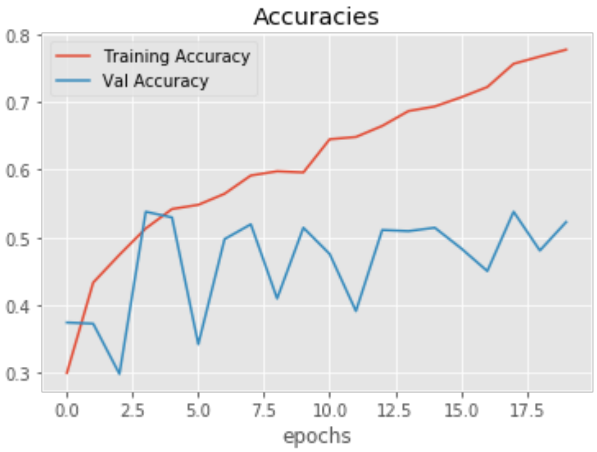


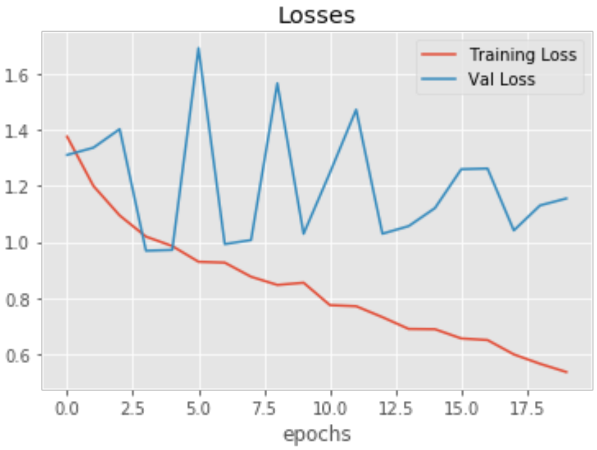
C-DRNN

In [31]:

```
from keras.layers import Input,Dense,concatenate,Flatten,GRU,Conv1D
from keras.models import Model

inputs= Input(shape=(t,f))

x = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(inputs)
x = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(x)
x = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(x)

res1 = GRU(32,activation='tanh',return_sequences=True)(x)
res2 = GRU(32,activation='tanh',return_sequences=True)(res1)

res1_2 = concatenate([res1,res2],axis=2)

res3 = GRU(32,activation='tanh',return_sequences=True)(res1_2)

x = concatenate([res1,res2,res3])

x = GRU(32,activation='tanh')(x)
predictions = Dense(4,activation='softmax')(x)

model = Model(inputs=inputs, outputs=predictions)

model.compile(optimizer = 'rmsprop',
             loss = 'categorical_crossentropy',
             metrics=['accuracy'])

#hist.history is a dictionary with all accs and losses
hist = model.fit(train_data_sliced,train_labels_sliced,epochs=20,validation
_split=0.25,batch_size=32,verbose=0)
test_score = model.evaluate(test_data_sliced, test_labels_sliced, batch_siz
e=32)

print "Test Results are ", test_score
plot_hist([hist.history['acc'],hist.history['val_acc']],['Training Accuracy
','Val Accuracy'],title='Accuracies')
plot_hist([hist.history['loss'],hist.history['val_loss']],['Training Loss',
'Val Loss'],title='Losses')
```

```
500/500 [==============================] - 1s 1ms/step
Test Results are  [1.1010976085662842, 0.522000000095367427]
```



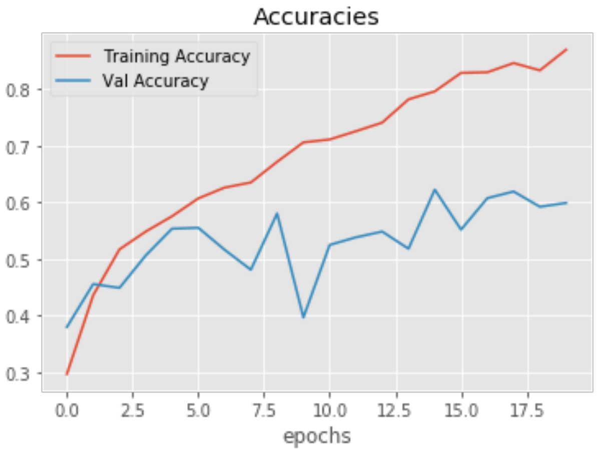


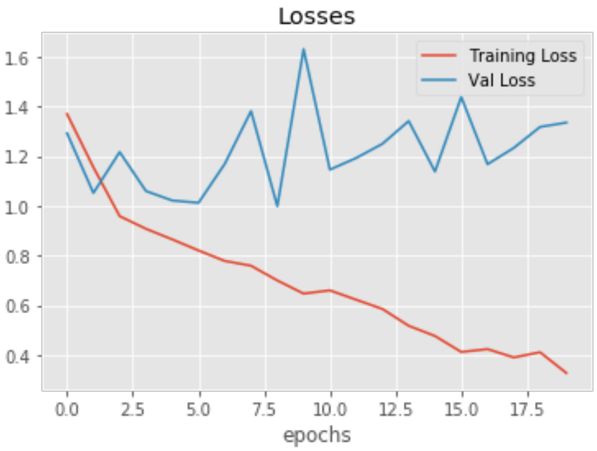
C-DRNN Testing

In [32]:

```
from keras.layers import Input,Dense,concatenate,Flatten,GRU,Conv1D,BatchNormalization,Dropout
from keras.models import Model

inputs= Input(shape=(t,f))

x = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(inputs)
x = BatchNormalization()(x)
x = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(x)
x = BatchNormalization()(x)
x = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(x)
x = Dropout(0.5)(x)

res1 = GRU(32,activation='tanh',return_sequences=True)(x)
res2 = GRU(32,activation='tanh',return_sequences=True)(res1)

res1_2 = concatenate([res1,res2],axis=2)

res3 = GRU(32,activation='tanh',return_sequences=True)(res2)

x = concatenate([res1,res2,res3])

x = GRU(32,activation='tanh')(x)
predictions = Dense(4,activation='softmax')(x)

model = Model(inputs=inputs, outputs=predictions)

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

#hist.history is a dictionary with all accs and losses
hist = model.fit(train_data_sliced,train_labels_sliced,epochs=20,validation_split=0.25,batch_size=32,verbose=0)
test_score = model.evaluate(test_data_sliced, test_labels_sliced, batch_size=32)

print "Test Results are ", test_score
plot_hist([hist.history['acc'],hist.history['val_acc']],['Training Accuracy','Val Accuracy'],title='Accuracies')
plot_hist([hist.history['loss'],hist.history['val_loss']],['Training Loss','Val Loss'],title='Losses')
```

```
500/500 [==============================] - 1s 1ms/step
Test Results are  [1.2032171669006348, 0.59200000047683721]
```



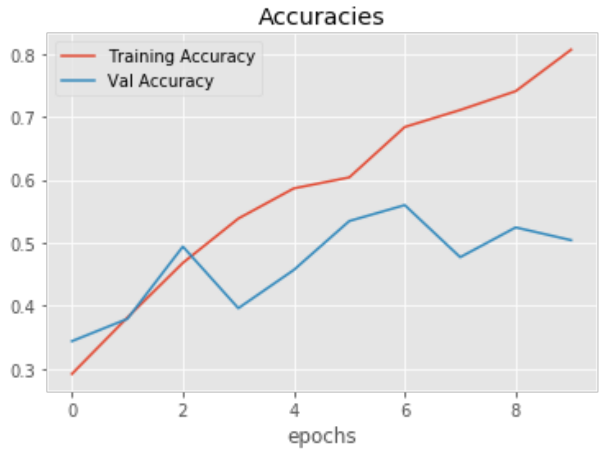


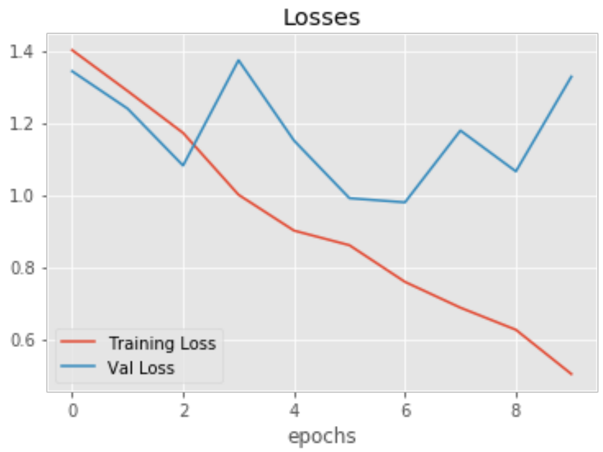
ChronoNet

In [35]:

```
from keras.layers import Input,Dense,concatenate,Flatten,GRU,Conv1D
from keras.models import Model

inputs= Input(shape=(t,f))

# First Inception
tower1 = Conv1D(32, 2, strides=2,activation='relu',padding="causal")(inputs
)
tower2 = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(inputs
)
tower3 = Conv1D(32, 8, strides=2,activation='relu',padding="causal")(inputs
)
x = concatenate([tower1,tower2,tower3],axis=2)

# Second Inception
tower1 = Conv1D(32, 2, strides=2,activation='relu',padding="causal")(x)
tower2 = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(x)
tower3 = Conv1D(32, 8, strides=2,activation='relu',padding="causal")(x)
x = concatenate([tower1,tower2,tower3],axis=2)

# Third Inception
tower1 = Conv1D(32, 2, strides=2,activation='relu',padding="causal")(x)
tower2 = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(x)
tower3 = Conv1D(32, 8, strides=2,activation='relu',padding="causal")(x)
x = concatenate([tower1,tower2,tower3],axis=2)

res1 = GRU(32,activation='tanh',return_sequences=True)(x)
res2 = GRU(32,activation='tanh',return_sequences=True)(res1)

res1_2 = concatenate([res1,res2],axis=2)

res3 = GRU(32,activation='tanh',return_sequences=True)(res2)

x = concatenate([res1,res2,res3])

x = GRU(32,activation='tanh')(x)
predictions = Dense(4,activation='softmax')(x)

model = Model(inputs=inputs, outputs=predictions)

model.compile(optimizer = 'rmsprop',
             loss = 'categorical_crossentropy',
             metrics=['accuracy'])

#hist.history is a dictionary with all accs and losses
hist = model.fit(train_data_sliced,train_labels_sliced,epochs=10,validation
_split=0.25,batch_size=64,verbose=0)
test_score = model.evaluate(test_data_sliced, test_labels_sliced, batch_siz
e=64)

print "Testing Accuracy is", test_score
plot_hist([hist.history['acc'],hist.history['val_acc']],['Training Accuracy
','Val Accuracy'],title='Accuracies')
plot_hist([hist.history['loss'],hist.history['val_loss']],['Training Loss',
'Val Loss'],title='Losses')
```

```
500/500 [==============================] - 1s 1ms/step
Testing Accuracy is [1.3355738878250123, 0.48599999761581419]
```

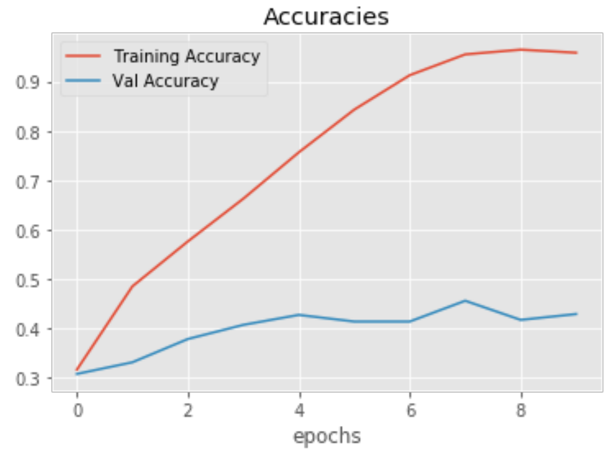


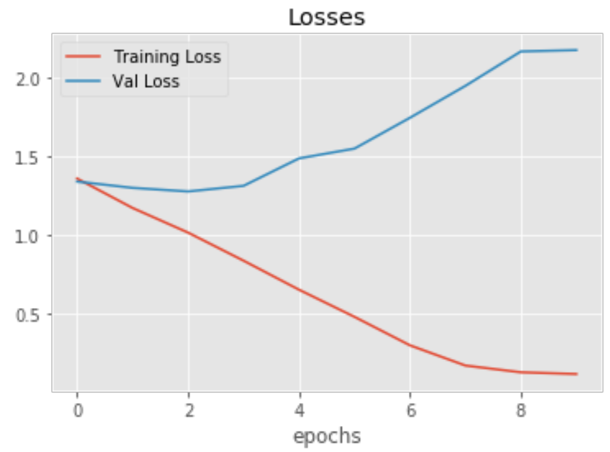


ChronoNet Model Testing

In [36]:

```
from keras.layers import Input,Dense,concatenate,Flatten,GRU,Conv1D
from keras.models import Model

inputs= Input(shape=(t,f))

# First Inception
tower1 = Conv1D(32, 2, strides=2,activation='relu',padding="causal")(inputs
)
tower1 = BatchNormalization()(tower1)
tower2 = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(inputs
)
tower2 = BatchNormalization()(tower2)
tower3 = Conv1D(32, 8, strides=2,activation='relu',padding="causal")(inputs
)
tower3 = BatchNormalization()(tower3)
x = concatenate([tower1,tower2,tower3],axis=2)

# Second Inception
tower1 = Conv1D(32, 2, strides=2,activation='relu',padding="causal")(x)
tower1 = BatchNormalization()(tower1)
tower2 = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(x)
tower2 = BatchNormalization()(tower2)
tower3 = Conv1D(32, 8, strides=2,activation='relu',padding="causal")(x)
tower3 = BatchNormalization()(tower3)
#x = concatenate([tower1,tower2,tower3],axis=2)

# Third Inception
tower1 = Conv1D(32, 2, strides=2,activation='relu',padding="causal")(x)
tower1 = BatchNormalization()(tower1)
tower2 = Conv1D(32, 4, strides=2,activation='relu',padding="causal")(x)
tower2 = BatchNormalization()(tower2)
tower3 = Conv1D(32, 8, strides=2,activation='relu',padding="causal")(x)
tower3 = BatchNormalization()(tower3)
x = concatenate([tower1,tower2,tower3],axis=2)

res1 = GRU(32,activation='tanh',return_sequences=True)(x)
res2 = GRU(32,activation='tanh',return_sequences=True)(res1)

res1_2 = concatenate([res1,res2],axis=2)

res3 = GRU(32,activation='tanh',return_sequences=True)(res2)

x = concatenate([res1,res2,res3])

x = GRU(32,activation='tanh')(x)
predictions = Dense(4,activation='softmax')(x)

model = Model(inputs=inputs, outputs=predictions)

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

#hist.history is a dictionary with all accs and losses
hist = model.fit(train_data_sliced,train_labels_sliced,epochs=10,validation
_split=0.25,batch_size=128,verbose=0)
test_score = model.evaluate(test_data_sliced, test_labels_sliced, batch_siz
e=128)

print "Testing Accuracy is", test_score
plot_hist([hist.history['acc'],hist.history['val_acc']],['Training Accuracy
','Val Accuracy'],title='Accuracies')
plot_hist([hist.history['loss'],hist.history['val_loss']],['Training Loss',
500/500 [==============================] - 1s 1ms/step
Testing Accuracy is [2.3090618019104006, 0.40599999666213987]
```


Accuracies
Training Accuracy
Val Accuracy
epochs


Losses
Training Loss
Val Loss
epochs