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

In [3]:
```
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 create_window_data
```

In [4]:
```
# vals is a list whose elements are the data values to be plotted
# labels is a list of labels for each respective data set
def plot_hist(vals,labels='Null',title='Null',xlabel='epochs'):
    n = len(vals)
    for i in range(n): plt.plot(vals[i],label=labels[i])
    plt.title(title)
    plt.xlabel(xlabel)
    plt.legend()
    plt.show()
```

In [5]:
```
A01T = h5py.File('datasets/A01T_slice.mat','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)
```

```
[[ 0.  0.  0.  1.]
 [ 0.  0.  1.  0.]
 [ 0.  1.  0.  0.]
 ...,
 [ 1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [ 1.  0.  0.  0.]]
```

In [6]:
```
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:]]
```

In [7]:
```
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
```

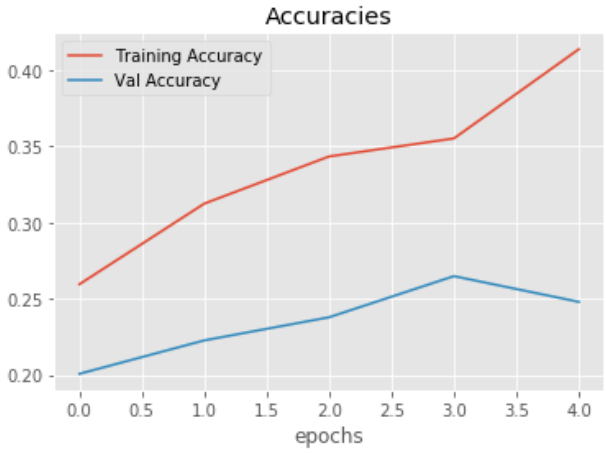
In [6]:
```
train_labels.shape
```

Out[6]: (2370, 4)

Everything from this point down is Testing

In [30]:
```
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=5,validation_split=0.25,bat
ch_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')
```

```
Train on 1777 samples, validate on 593 samples
Epoch 1/5
1777/1777 [==============================] - 35s 20ms/step - loss: 1.3932 -
acc: 0.2594 - val_loss: 1.4080 - val_acc: 0.2007
Epoch 2/5
1777/1777 [==============================] - 33s 19ms/step - loss: 1.3707 -
acc: 0.3123 - val_loss: 1.4014 - val_acc: 0.2226
Epoch 3/5
1777/1777 [==============================] - 31s 17ms/step - loss: 1.3518 -
acc: 0.3433 - val_loss: 1.4275 - val_acc: 0.2378
Epoch 4/5
1777/1777 [==============================] - 30s 17ms/step - loss: 1.3238 -
acc: 0.3551 - val_loss: 1.4054 - val_acc: 0.2648
Epoch 5/5
1777/1777 [==============================] - 31s 18ms/step - loss: 1.2820 -
acc: 0.4136 - val_loss: 1.4213 - val_acc: 0.2479
500/500 [==============================] - 3s 5ms/step
[1.4121294736862182, 0.28200000023841859]
```



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 [27]:
```
# 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

#del model
model = Sequential()
#model.add(LSTM(100, input_shape=(1000, 22)))
model.add(Conv1D(32, 3, activation='relu',input_shape=(t,f)))            #
Originally 32 each
model.add(BatchNormalization())
model.add(Conv1D(32, 3, activation='relu'))
model.add(MaxPooling1D())
model.add(Dropout(0.25))

model.add(Conv1D(64, 3, activation='relu'))
#Originally 64 each
model.add(BatchNormalization())
model.add(Conv1D(64, 3, 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)

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

hist = model.fit(train_data,train_labels,epochs=20,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')
```

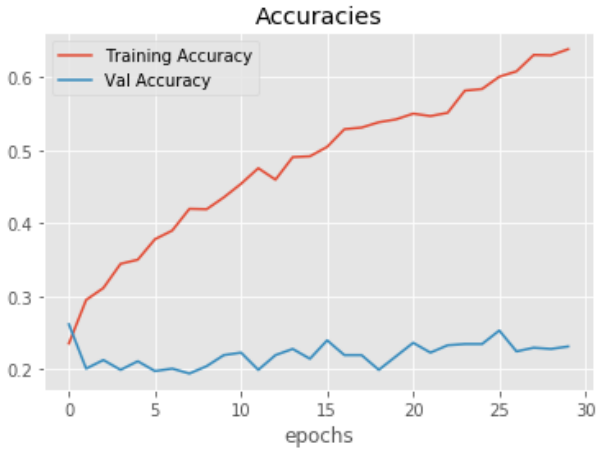
```
500/500 [==============================] - 1s 2ms/step
```

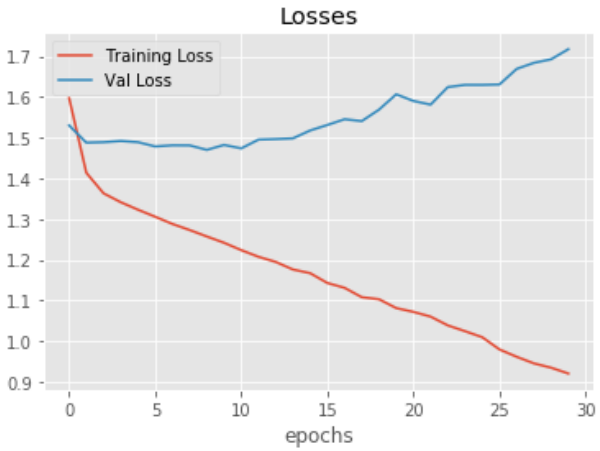
Simple RNN model

In [21]:
```
model = Sequential([
    SimpleRNN(64, input_shape=(t,f)),
    Dense(32),
    BatchNormalization(),
    Activation('relu'),
    Dense(4),
    Activation('softmax'),
])

model.compile(optimizer = 'adam',
              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)
```

```
500/500 [==============================] - 1s 2ms/step
```

In [22]:
```
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')
```





CHRONONET PAPER

Here is the implementation of Figure 1b based on ChronoNet paper

In [16]:
```
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='relu',return_sequences=True))
model.add(GRU(32,activation='relu',return_sequences=True))
model.add(GRU(32,activation='relu'))


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,train_labels,epochs=10,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')
```

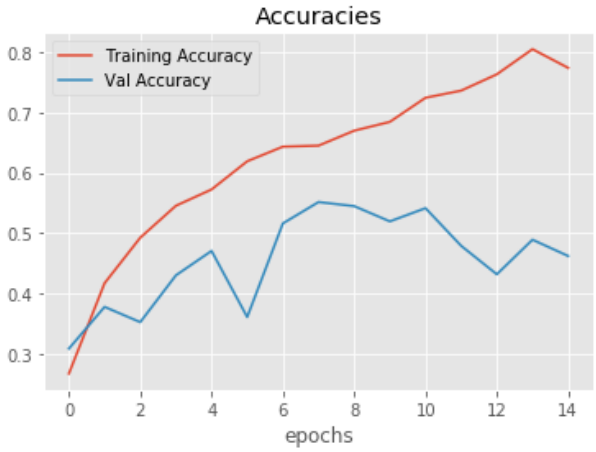
```
500/500 [==============================] - 1s 2ms/step
[1.0793222360610961, 0.54200000047683716]
```

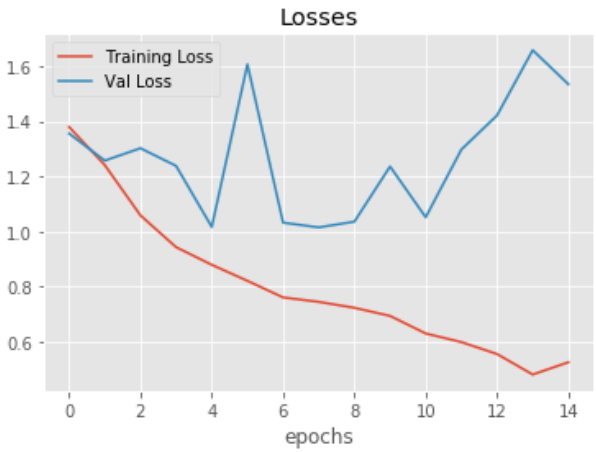
Implementation of Figure 1b but adding regularization structures like that found in VGGnet

In [17]:
```
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'))

#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=['accu
racy'])

#model.add()

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

#hist.history is a dictionary with all accs and losses
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)
```

```
WARNING:tensorflow:Variable *= will be deprecated. Use variable.assign_mul
if you want assignment to the variable value or 'x = x * y' if you want a n
ew python Tensor object.
500/500 [==============================] - 1s 1ms/step
[1.0582130537033081, 0.60799999999999998]
```

In [18]:
```
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')
```





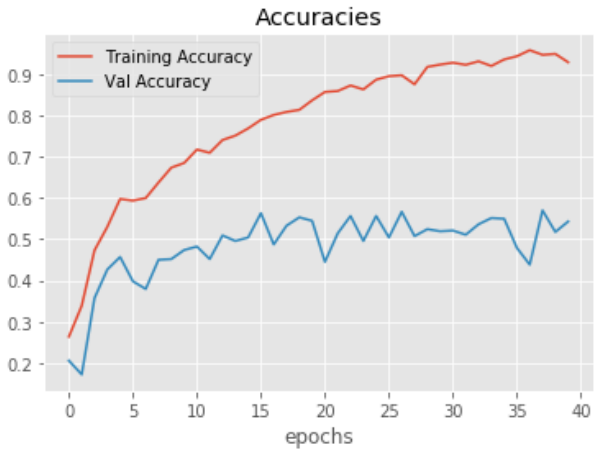
Replaced GRU with LSTM

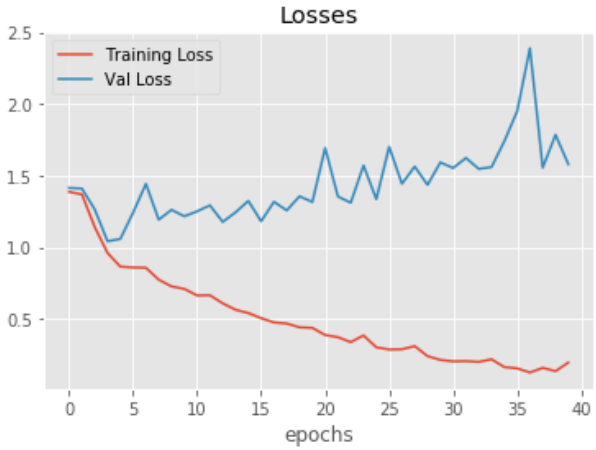
In [19]:
```
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv1D, MaxPooling1D, BatchNormalization,GRU,LSTM
from keras.optimizers import SGD

#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'))

#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=['accu
racy'])

#model.add()

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

#hist.history is a dictionary with all accs and losses
hist = model.fit(train_data,train_labels,epochs=40,validation_split=0.25,ba
tch_size=32,verbose=0)
test_score = model.evaluate(test_data, test_labels, batch_size=32)
print(test_score)
```

```
500/500 [==============================] - 1s 1ms/step
[1.3234160480499269, 0.60800000011920929]
```

In [20]:
```
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')
```

```
Test Results are  [1.3234160480499269, 0.60800000011920929]
```





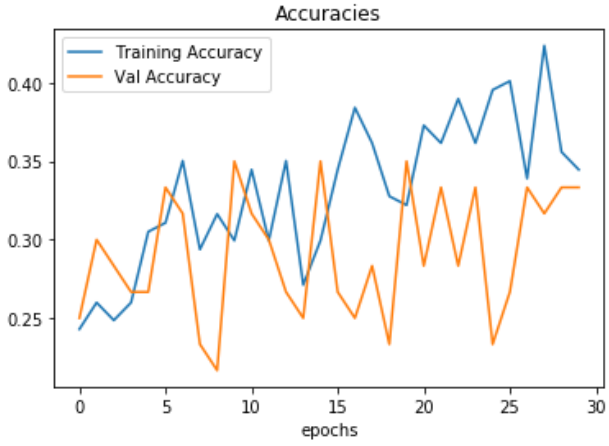
IC-RNN

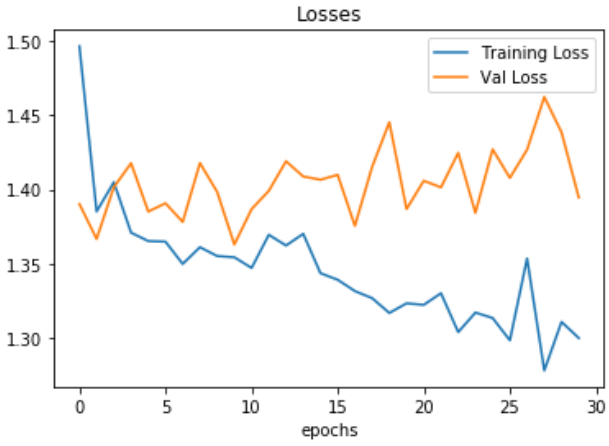
In [42]:
```
from keras.layers import Input,Dense,concatenate,Flatten,GRU,Conv1D
from keras.models import Model

inputs= Input(shape=(1000,22))

# 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')(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,train_labels,epochs=30,validation_split=0.25,ba
tch_size=64,verbose=0)
test_score = model.evaluate(test_data, test_labels, batch_size=64)
```

```
50/50 [==============================] - 0s 4ms/step
```

In [43]:
```
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')
```

```
Test Results are  [1.4721500873565674, 0.2199999988079071]
```





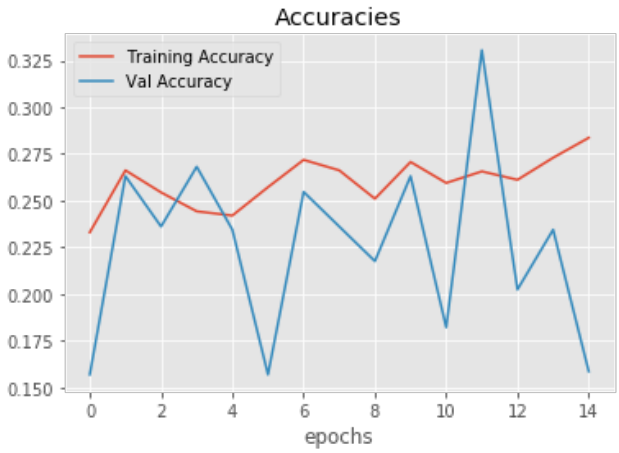
IC-RNN Testing

In [19]:
```
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')(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,train_labels,epochs=15,validation_split=0.25,ba
tch_size=64,verbose=0)
test_score = model.evaluate(test_data, test_labels, batch_size=64)
```

```
500/500 [==============================] - 2s 3ms/step
```

In [20]:
```
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')
```

```
Test Results are  [1.387840368270874, 0.28400000023841859]
```



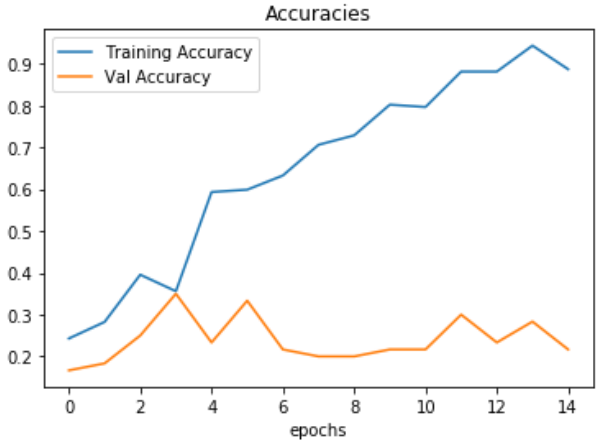
C-DRNN

In [8]:
```
from keras.layers import Input,Dense,concatenate,Flatten,GRU,Conv1D
from keras.models import Model

inputs= Input(shape=(1000,22))

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,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)
```

```
50/50 [==============================] - 0s 4ms/step
```

In [9]:
```
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')
```

```
Test Results are  [1.8914880275726318, 0.30000000238418578]
```



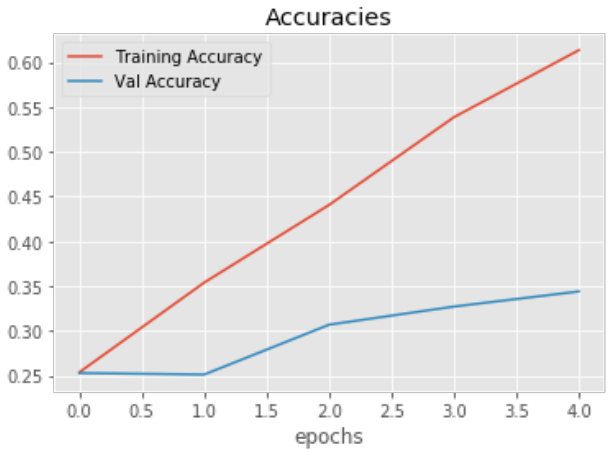
C-DRNN Testing

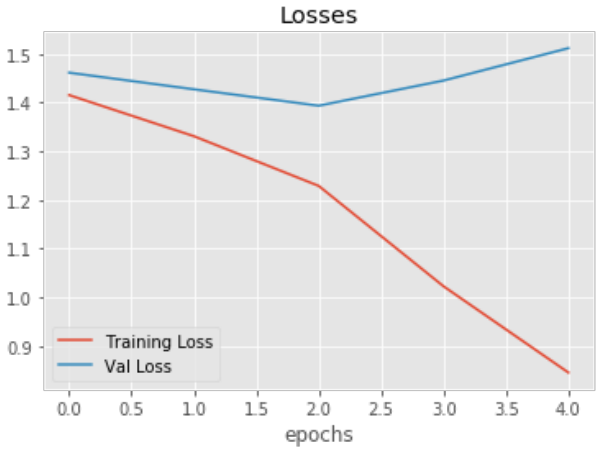
In [21]:
```
from keras.layers import Input,Dense,concatenate,Flatten,GRU,Conv1D,BatchNo
rmalization,Dropout
from keras.models import Model

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,train_labels,epochs=5,validation_split=0.25,bat
ch_size=32,verbose=0)
test_score = model.evaluate(test_data, test_labels, batch_size=32)
```

```
500/500 [==============================] - 3s 7ms/step
```

In [22]:
```
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')
```

```
Test Results are  [1.3403036570549012, 0.40599999952316285]
```





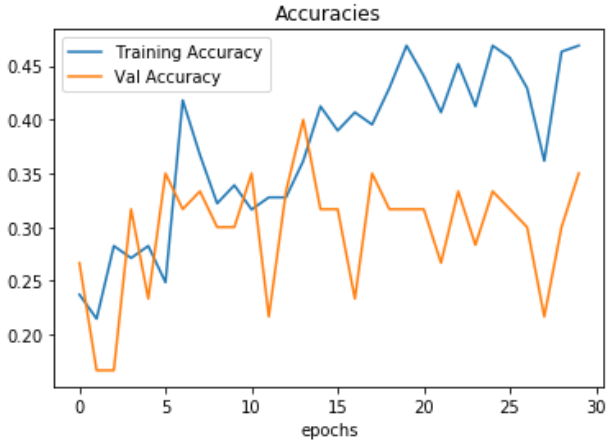
ChronoNet

In [49]:
```
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,train_labels,epochs=30,validation_split=0.25,ba
tch_size=32,verbose=0)
test_score = model.evaluate(test_data, test_labels, batch_size=32)
```

```
50/50 [==============================] - 0s 7ms/step
```

In [50]:
```
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')
```

```
Testing Accuracy is [1.6881582021713257, 0.2600000047683716]
```



ChronoNet Model Testing

In [28]:
```
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,train_labels,epochs=10,validation_split=0.25,ba
tch_size=32,verbose=0)
test_score = model.evaluate(test_data, test_labels, batch_size=32)

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 [==============================] - 3s 6ms/step
Testing Accuracy is [1.8246414136886597, 0.47999999952316286]
```