In [1]:

```
from keras.datasets import mnist
from keras import models
from keras import layers
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

```
D:\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion o
f the second argument of issubdtype from `float` to `np.floating` is deprecate
d. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [2]:

```
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255

from keras.utils import to_categorical

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

In [3]:

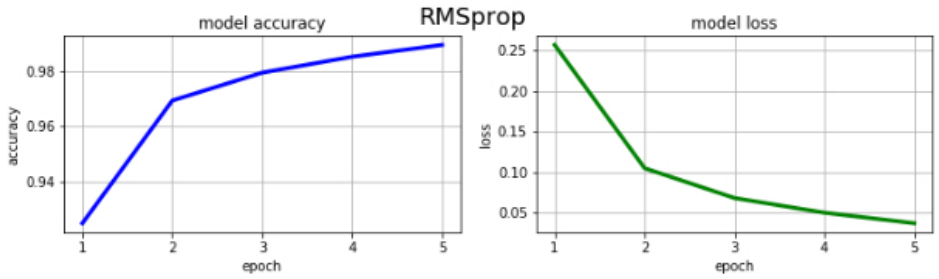
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:

```
#优化器rmsprop
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics
rmsprop_hist = network.fit(train_images, train_labels, epochs=5, batch_size=
rmsprop_test_loss, rmsprop_test_acc = network.evaluate(test_images, test_labe
print('test_loss:', rmsprop_test_loss)
print('test_acc:', rmsprop_test_acc)
print(rmsprop_hist.history)

plt.figure(figsize=(10,3))
plt.suptitle('RMSprop', fontsize=18)

plt.subplot(1,2,1)
plt.plot(rmsprop_hist.history['acc'], 'b', linewidth=3)
plt.title('model accuracy')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.grid(True)
plt.tight_layout()

plt.subplot(1,2,2)
plt.plot(rmsprop_hist.history['loss'], 'g', linewidth=3)
plt.title('model loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.grid(True)
plt.tight_layout()

plt.savefig('Accuracy of RMSprop.jpg',dpi=300)
```

```
WARNING:tensorflow:From D:\Anaconda3\lib\site-packages\tensorflow\python\framew
ork\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops)
is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\Anaconda3\lib\site-packages\tensorflow\python\ops\ma
th_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated an
d will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/5
```

```
60000/60000 [==============================] - 9s 152us/step - loss: 0.2565 - a
cc: 0.9248
Epoch 2/5
60000/60000 [==============================] - 3s 52us/step - loss: 0.1048 - ac
c: 0.9692
Epoch 3/5
60000/60000 [==============================] - 3s 53us/step - loss: 0.0681 - ac
c: 0.9793
Epoch 4/5
60000/60000 [==============================] - 3s 51us/step - loss: 0.0500 - ac
c: 0.9851
Epoch 5/5
60000/60000 [==============================] - 3s 55us/step - loss: 0.0371 - ac
c: 0.9894
10000/10000 [==============================] - 1s 59us/step
test_loss: 0.06734216312763747
test_acc: 0.9799
{'loss': [0.2565254210035006, 0.10480482665896415, 0.06810588288803895, 0.05004
766948421796, 0.03708880497068167], 'acc': [0.9247833333333333, 0.9692333333015
442, 0.9793333333333333, 0.9851000000317891, 0.9893833333333333]}
```

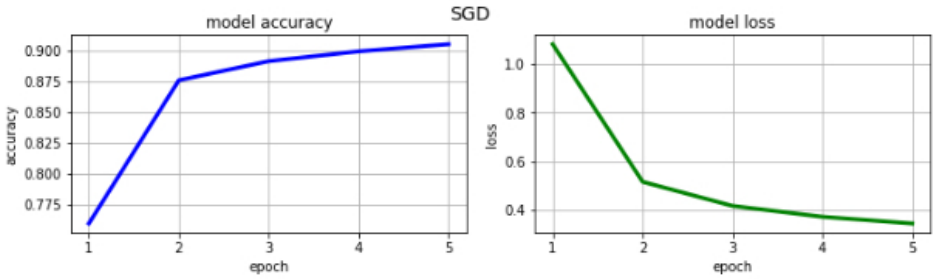


In [5]:

```python
#優化器sgd
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['a
sgd_hist = network.fit(train_images, train_labels, epochs=5, batch_size=128)
sgd_test_loss, sgd_test_acc = network.evaluate(test_images, test_labels)
print('test_loss:', sgd_test_loss)
print('test_acc:', sgd_test_acc)
print(sgd_hist.history)

plt.figure(figsize=(10,3))
plt.suptitle('SGD', fontsize=14)

plt.subplot(1,2,1)
plt.plot(sgd_hist.history['acc'], 'b', linewidth=3)
plt.title('model accuracy')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.grid(True)
plt.tight_layout()

plt.subplot(1,2,2)
plt.plot(sgd_hist.history['loss'], 'g', linewidth=3)
plt.title('model loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.grid(True)
plt.tight_layout()

plt.savefig('Accuracy of SGD.jpg',dpi=300)
```

```
Epoch 1/5
60000/60000 [==============================] - 3s 45us/step - loss: 1.0830 - ac
c: 0.7597
Epoch 2/5
60000/60000 [==============================] - 3s 44us/step - loss: 0.5148 - ac
c: 0.8761
Epoch 3/5
60000/60000 [==============================] - 3s 42us/step - loss: 0.4152 - ac
c: 0.8916
Epoch 4/5
60000/60000 [==============================] - 3s 42us/step - loss: 0.3703 - ac
c: 0.8996
Epoch 5/5
60000/60000 [==============================] - 3s 45us/step - loss: 0.3427 - ac
```

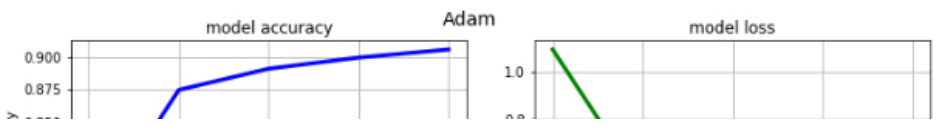
```
c: 0.9054
10000/10000 [==============================] - 1s 62us/step
test_loss: 0.31555847700238226
test_acc: 0.9127
{'loss': [1.0829915905316672, 0.5147729955196381, 0.41524283391634625, 0.370302
22096443177, 0.3426549215475718], 'acc': [0.7596999999682108, 0.876050000031789
1, 0.8915833333651225, 0.8995500000317892, 0.9053833333333333]}
```

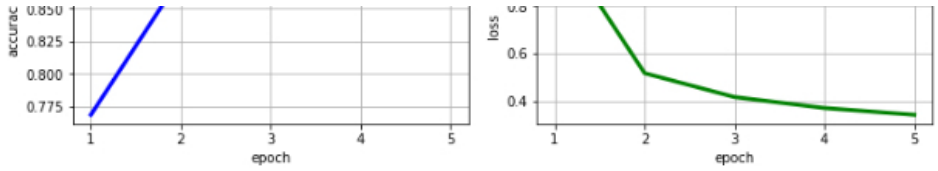


In [6]:

```
#优化器adam
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['a
adam_hist = network.fit(train_images, train_labels, epochs=5, batch_size=128)
adam_test_loss, adam_test_acc = network.evaluate(test_images, test_labels)
print('test_loss:', adam_test_loss)
print('test_acc:', adam_test_acc)
print(adam_hist.history)

plt.figure(figsize=(10,3))
plt.suptitle('Adam', fontsize=14)

plt.subplot(1,2,1)
plt.plot(adam_hist.history['acc'], 'b', linewidth=3)
plt.title('model accuracy')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.grid(True)
plt.tight_layout()

plt.subplot(1,2,2)
plt.plot(adam_hist.history['loss'], 'g', linewidth=3)
plt.title('model loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.grid(True)
plt.tight_layout()

plt.savefig('Accuracy of Adam.jpg',dpi=300)
```

```
Epoch 1/5
60000/60000 [==============================] - 4s 63us/step - loss: 1.0923 - ac
c: 0.7684
Epoch 2/5
60000/60000 [==============================] - 3s 42us/step - loss: 0.5180 - ac
c: 0.8746
Epoch 3/5
60000/60000 [==============================] - 3s 42us/step - loss: 0.4177 - ac
c: 0.8909
Epoch 4/5
60000/60000 [==============================] - 3s 42us/step - loss: 0.3716 - ac
c: 0.8995
Epoch 5/5
60000/60000 [==============================] - 3s 42us/step - loss: 0.3430 - ac
c: 0.9057
10000/10000 [==============================] - 1s 61us/step
test_loss: 0.31735005323290827
test_acc: 0.9153
{'loss': [1.0923182662963866, 0.5180033554871877, 0.4176862187862396, 0.3715899
117946625, 0.34304365968704226], 'acc': [0.7683500000317891, 0.874616666698455
8, 0.89095, 0.8995333333015442, 0.9057499999682108]}
```

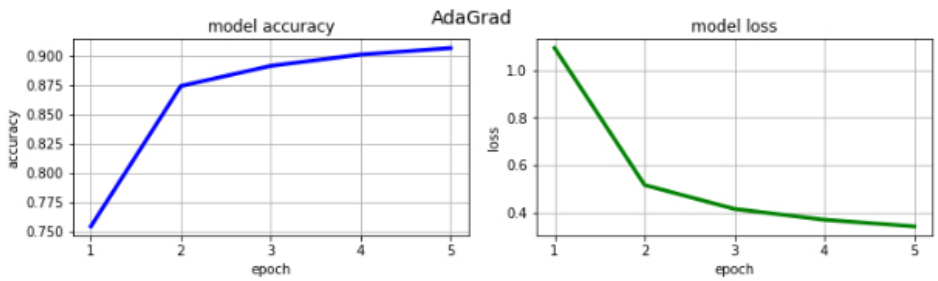




In [7]:

```
#优化器adagrad
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['a
adagrad_hist = network.fit(train_images, train_labels, epochs=5, batch_size=1
adagrad_test_loss, adagrad_test_acc = network.evaluate(test_images, test_labe
print('test_loss:', adagrad_test_loss)
print('test_acc:', adagrad_test_acc)
print(adagrad_hist.history)

plt.figure(figsize=(10,3))
plt.suptitle('AdaGrad', fontsize=14)

plt.subplot(1,2,1)
plt.plot(adagrad_hist.history['acc'], 'b', linewidth=3)
plt.title('model accuracy')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.grid(True)
plt.tight_layout()

plt.subplot(1,2,2)
plt.plot(adagrad_hist.history['loss'], 'g', linewidth=3)
plt.title('model loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.grid(True)
plt.tight_layout()

plt.savefig('Accuracy of AdaGrad.jpg',dpi=300)
```

```
Epoch 1/5
60000/60000 [==============================] - 3s 47us/step - loss: 1.0931 - ac
c: 0.7546
Epoch 2/5
60000/60000 [==============================] - 3s 43us/step - loss: 0.5155 - ac
c: 0.8742
Epoch 3/5
60000/60000 [==============================] - 3s 48us/step - loss: 0.4146 - ac
c: 0.8913
Epoch 4/5
60000/60000 [==============================] - 3s 42us/step - loss: 0.3690 - ac
c: 0.9009
Epoch 5/5
60000/60000 [==============================] - 3s 44us/step - loss: 0.3410 - ac
c: 0.9065
10000/10000 [==============================] - 1s 61us/step
test_loss: 0.31547285937070846
test_acc: 0.9154
{'loss': [1.0930681081136067, 0.5154599870204926, 0.4146225591023763, 0.3690103
939851125, 0.3410487915198083], 'acc': [0.7546166666348775, 0.87415, 0.8913166
666666666, 0.9009333333333334, 0.9064833333015442]}
```

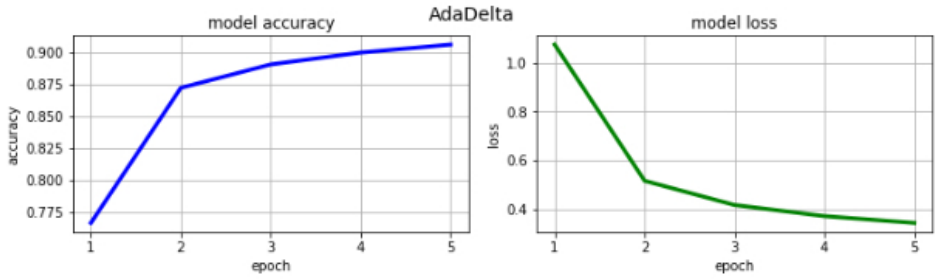


In [8]:

```
#优化器adadelta
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['a
```

```
adadelta_hist = network.fit(train_images, train_labels, epochs=5, batch_size=
adadelta_test_loss, adadelta_test_acc = network.evaluate(test_images, test_la
print('test_loss:', adadelta_test_loss)
print('test_acc:', adadelta_test_acc)
print(adadelta_hist.history)

plt.figure(figsize=(10,3))
plt.suptitle('AdaDelta', fontsize=14)

plt.subplot(1,2,1)
plt.plot(adadelta_hist.history['acc'], 'b', linewidth=3)
plt.title('model accuracy')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.grid(True)
plt.tight_layout()

plt.subplot(1,2,2)
plt.plot(adadelta_hist.history['loss'], 'g', linewidth=3)
plt.title('model loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.grid(True)
plt.tight_layout()

plt.savefig('Accuracy of AdaDelta.jpg',dpi=300)
```

```
Epoch 1/5
60000/60000 [==============================] - 3s 49us/step - loss: 1.0738 - ac
c: 0.7666
Epoch 2/5
60000/60000 [==============================] - 3s 44us/step - loss: 0.5171 - ac
c: 0.8720
Epoch 3/5
60000/60000 [==============================] - 3s 43us/step - loss: 0.4181 - ac
c: 0.8902
Epoch 4/5
60000/60000 [==============================] - 3s 43us/step - loss: 0.3727 - ac
c: 0.8995
Epoch 5/5
60000/60000 [==============================] - 3s 43us/step - loss: 0.3448 - ac
c: 0.9057
10000/10000 [==============================] - 1s 64us/step
test_loss: 0.31858390017151833
test_acc: 0.9103
{'loss': [1.0738442731539408, 0.5170751467227935, 0.4181419246514638, 0.3726661
7852846784, 0.34476230732599894], 'acc': [0.7665666666984559, 0.871966666698455
8, 0.8902499999682109, 0.8994999999682108, 0.9057]}
```

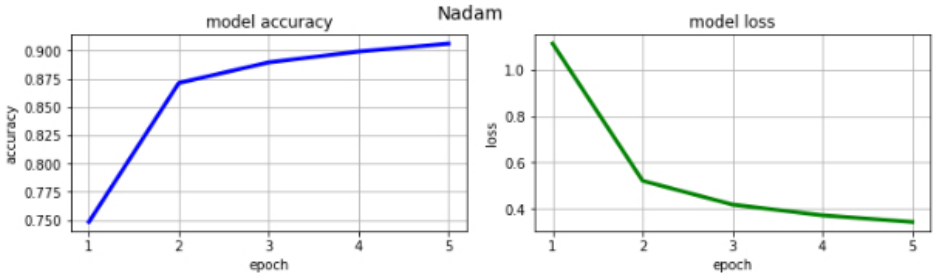


In [9]:

```
#优化器nadam
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['
nadam_hist = network.fit(train_images, train_labels, epochs=5, batch_size=128
nadam_test_loss, nadam_test_acc = network.evaluate(test_images, test_labels)
print('test_loss:', nadam_test_loss)
print('test_acc:', nadam_test_acc)
print(nadam_hist.history)

plt.figure(figsize=(10,3))
plt.suptitle('Nadam', fontsize=14)

plt.subplot(1,2,1)
plt.plot(nadam_hist.history['acc'], 'b', linewidth=3)
plt.title('model accuracy')
```

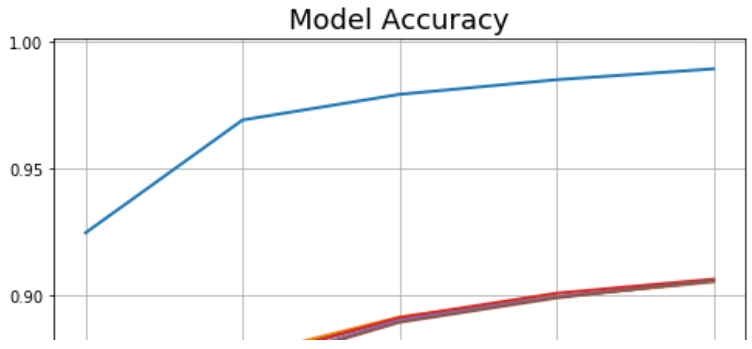
```
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.grid(True)
plt.tight_layout()

plt.subplot(1,2,2)
plt.plot(nadam_hist.history['loss'], 'g', linewidth=3)
plt.title('model loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.grid(True)
plt.tight_layout()

plt.savefig('Accuracy of Nadam.jpg',dpi=300)
```

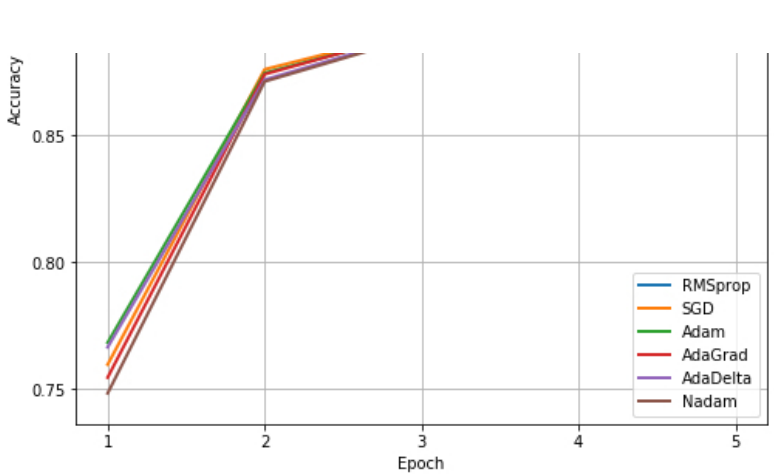
```
Epoch 1/5
60000/60000 [==============================] - 3s 46us/step - loss: 1.1132 - ac
c: 0.7484
Epoch 2/5
60000/60000 [==============================] - 3s 46us/step - loss: 0.5214 - ac
c: 0.8713
Epoch 3/5
60000/60000 [==============================] - 3s 45us/step - loss: 0.4187 - ac
c: 0.8895
Epoch 4/5
60000/60000 [==============================] - 2s 41us/step - loss: 0.3723 - ac
c: 0.8991
Epoch 5/5
60000/60000 [==============================] - 3s 44us/step - loss: 0.3436 - ac
c: 0.9061
10000/10000 [==============================] - 1s 63us/step
test_loss: 0.31700938050150873
test_acc: 0.913
{'loss': [1.113154947121938, 0.5213805438995361, 0.4187394819736481, 0.37233679
865201313, 0.3436127088069916], 'acc': [0.7483833333651225, 0.8712833333015442,
0.8895166666984559, 0.8990999999682109, 0.9060666666348776]}
```



In [10]:

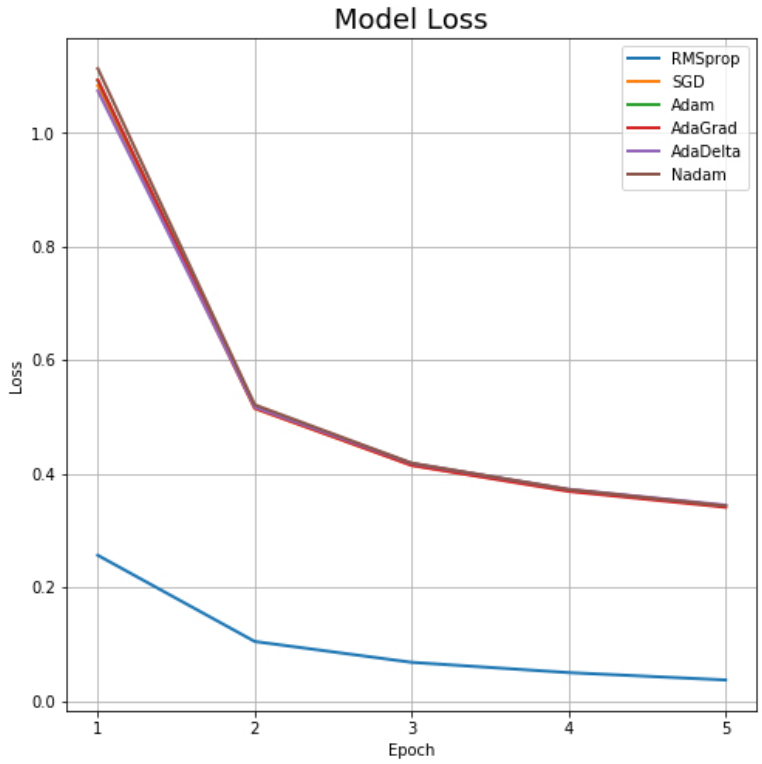
```
# summarize history for accuracy
plt.figure(figsize=(8,8))
plt.plot(rmsprop_hist.history['acc'], linewidth=2)
plt.plot(sgd_hist.history['acc'], linewidth=2)
plt.plot(adam_hist.history['acc'], linewidth=2)
plt.plot(adagrad_hist.history['acc'], linewidth=2)
plt.plot(adadelta_hist.history['acc'], linewidth=2)
plt.plot(nadam_hist.history['acc'], linewidth=2)
plt.title('Model Accuracy', fontsize=18)
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.legend(['RMSprop', 'SGD', 'Adam', 'AdaGrad', 'AdaDelta', 'Nadam'], loc='b
plt.grid(True)
plt.savefig('Accuracy of different optimizer.jpg',dpi=300)
```





In [11]:

```
# summarize history for loss
plt.figure(figsize=(8,8))
plt.plot(rmsprop_hist.history['loss'], linewidth=2)
plt.plot(sgd_hist.history['loss'], linewidth=2)
plt.plot(adam_hist.history['loss'], linewidth=2)
plt.plot(adagrad_hist.history['loss'], linewidth=2)
plt.plot(adadelta_hist.history['loss'], linewidth=2)
plt.plot(nadam_hist.history['loss'], linewidth=2)
plt.title('Model Loss', fontsize=18)
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.xticks([0, 1, 2, 3, 4], ['1', '2', '3', '4', '5'])
plt.legend(['RMSprop', 'SGD', 'Adam', 'AdaGrad', 'AdaDelta', 'Nadam'], loc='
plt.grid(True)
plt.savefig('Loss of different optimizer.jpg',dpi=300)
```



In [ ]:

```
1
```