Apostolos Papatheodorou

> *TypeSQL: A Knowledge-Based Type-Aware Text-to-SQL Model.*

In [20]:
```python
import time
import numpy as np
import matplotlib.pyplot as plt

EPOCHS=7
```

This project concerns the examinination and implementaion of three NLP to SQL models

- TypeSQL
- SQLNet
- Seq2SQl

& it is splited in two parts

1. Conclusions
2. Project details

> In part 1. you can see the most important results of the experiments.
> For more details check part 2.

How to Run this notepad?

> **It is recommended, before studying this notepad, to go to the last cell of the file and to run it. This cell contains all the fuzzy methods for plots which are used.**
>
> **Afterwards you can examine and check all the project resutls from wherever you want.**

1. Conclusions

After the training and testing of the models we take the following results.

DEV ACCURACY (epochs=7)

Model	Agg	Select	Where
Seq2SQL	92.3%	84.5%	44.4%
SQLNet-Col.Atten.	89.8%	89.0%	53.8%
SQLNet-Col.Atten-Embed.	89.8%	.91.2%	62.9%
TypeSQL-Cont.0.	**89.8%**	**92.5%**	**75.5%**
TypeSQL-Cont.1	**90.0%**	**91.9%**	**86.7%**

Test ACCURACY (epochs=7)

Model	Agg	Select	Where
Seq2SQL	92.3%	84.3%	45.0%
SQLNet-Col.Atten.	90.0%	88.0%	53.4%
SQLNet-Col.Atten-Embed.	90.0%	90.0%	62.4%
TypeSQL-Cont.0	**89.9%**	**92.1%**	**75.8%**
TypeSQL-Cont.1	**90.0%**	**91.9%**	**86.7%**

Notice 1

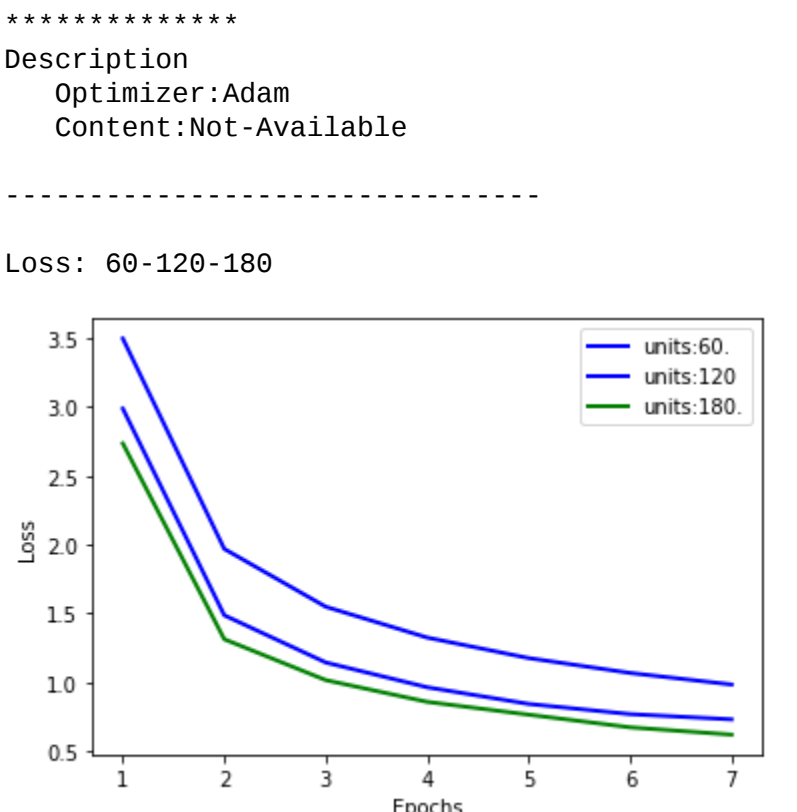
> For TypeSQL with available content and for units=180 we have dev and test accuracy 86.7% which is an improvement in comparison with the author's model (85.5% and 82.6% respectively). This change is due to the increase of the LSTMs units > (180 instead of 120).

Notice 2

> *From the above diagrams we observe than the Seq2SQL model has different behavior from the expected (we achieved 45% accuracy while it is expected to be around to 60%). To determine if this outcome comes from the limited number of epochs (ep==7) or from implementaion we plot the epoch/acc. the where-diagram.*

In [21]:
```python
Seq2SQL_Where_Acc()
```

```
Description: Seq2SQL-Where TRAIN-VALIDATION
    Epoch_time: 9.65 min
    Loss:[3.541 2.305 1.785 1.522 1.353 1.235 1.14]
    Dev   [0.154, 0.247, 0.3, 0.357, 0.387, 0.415, 0.444]
    train [0.165, 0.287, 0.359, 0.421, 0.458, 0.497, 0.527]
--------------------------------
```



Indeed it is obserbed that due to the small No of epochs the models have not converge yet and therefore further iterations are demanded.

In [ ]:

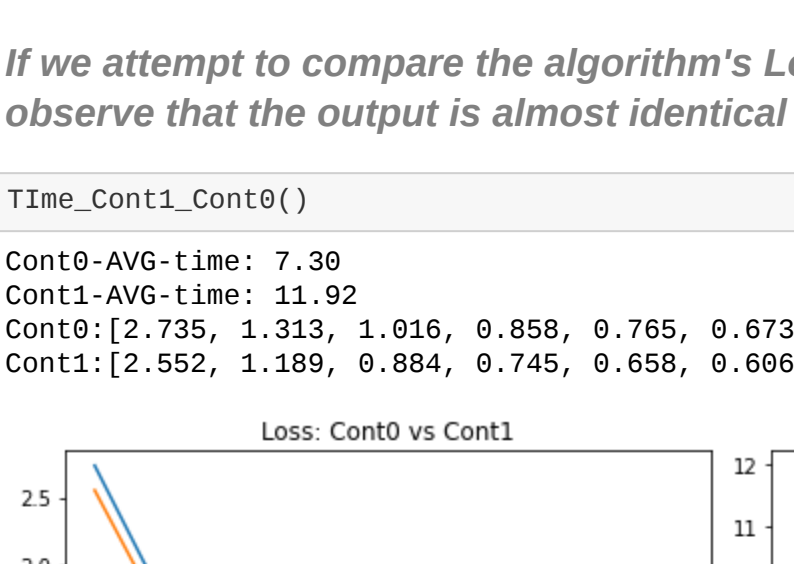
2. Project Details

In [59]:
```python
Units_60_120_180_ContNo()
```

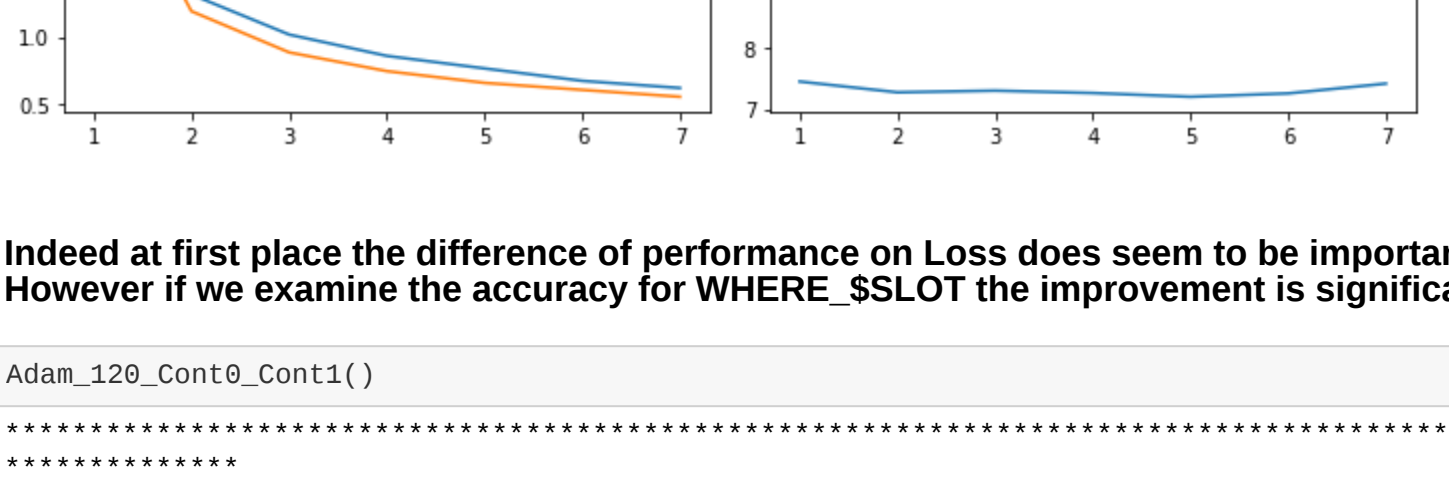
```
****************************************************************************************************
Description
   Optimizer:Adam
   Content:Not-Available

--------------------------------

Loss: 60-120-180
```



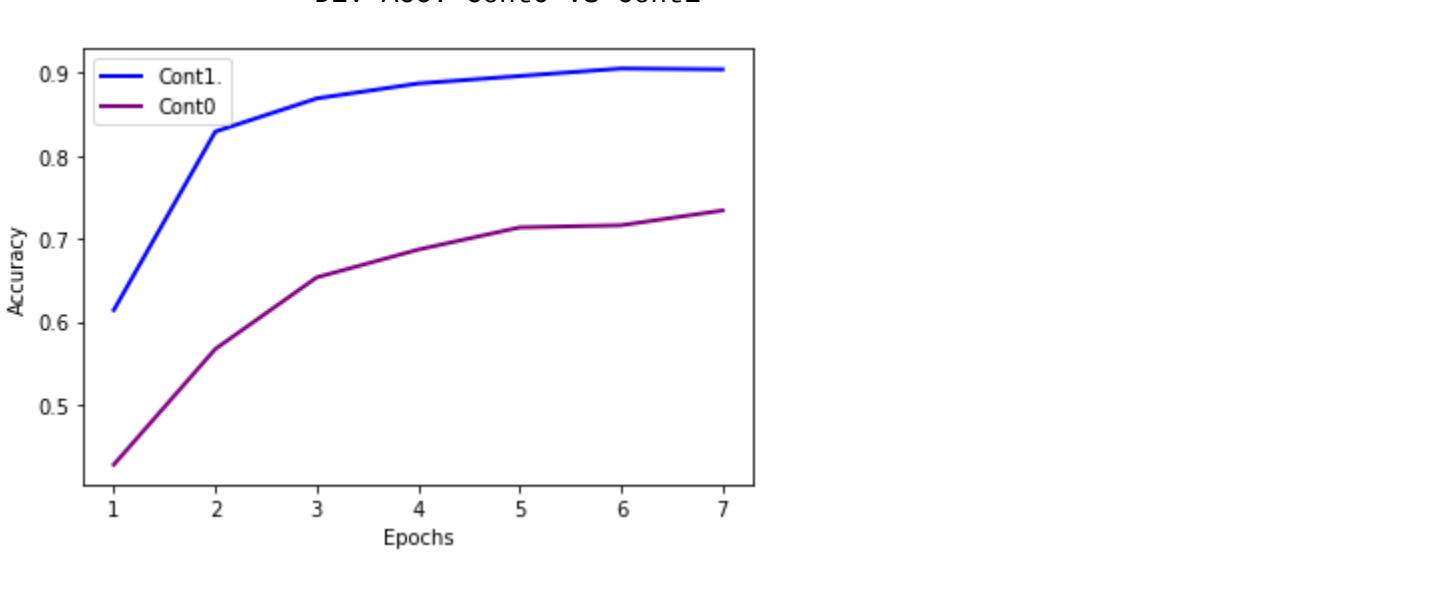

```
    valAcc_60: [0.189, 0.373, 0.445, 0.476, 0.506, 0.531, 0.553]
    valAcc_120 [0.361, 0.516, 0.567, 0.589, 0.607, 0.618, 0.618]
    valAcc_180 [0.449, 0.553, 0.594, 0.617, 0.624, 0.638, 0.646]
```



If we attempt to compare the algorithm's Loss with and without content we will observe that the output is almost identical

In [30]:
```python
TIme_Cont1_Cont0()
```

```
Cont0-AVG-time: 7.30
Cont1-AVG-time: 11.92
Cont0:[2.735, 1.313, 1.016, 0.858, 0.765, 0.673, 0.62 ]
Cont1:[2.552, 1.189, 0.884, 0.745, 0.658, 0.606, 0.555]
```



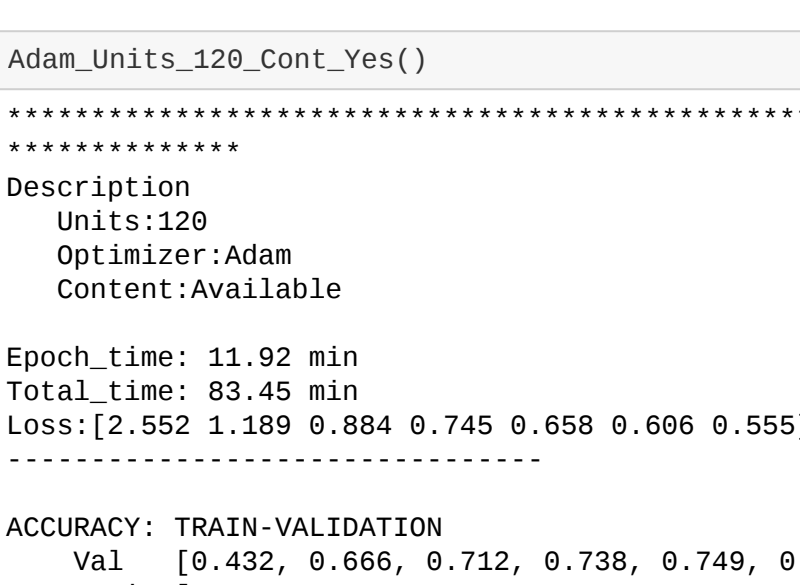
Indeed at first place the difference of performance on Loss does seem to be important. However if we examine the accuracy for WHERE_$SLOT the improvement is significant.

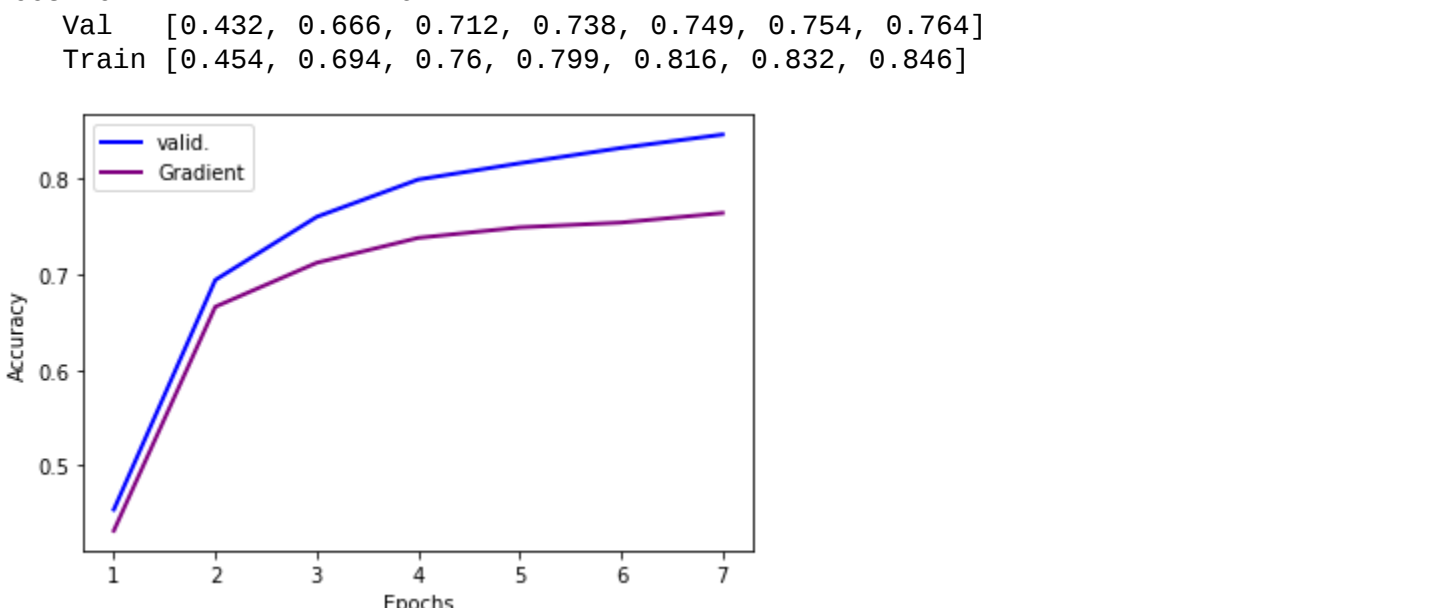
In [44]:
```python
Adam_120_Cont0_Cont1()
```

```
****************************************************************************************************
Description Cont:0 vs Cont:1
   Units:120
   Optimizer:Adam

   Cont0: Avg_time=7.37 min
   Cont1: Avg_time=11.92 min
--------------------------------

                    DEV-ACC: Cont0 VS Cont1
```





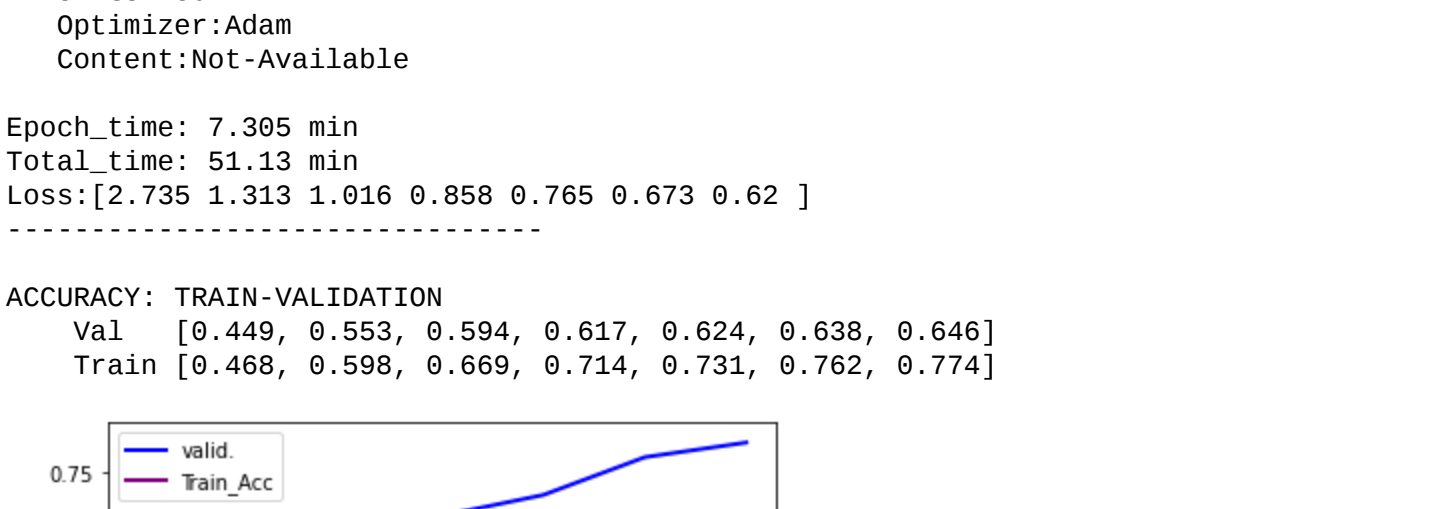
By further analizing the performance of model with Adam optimizer, we have:

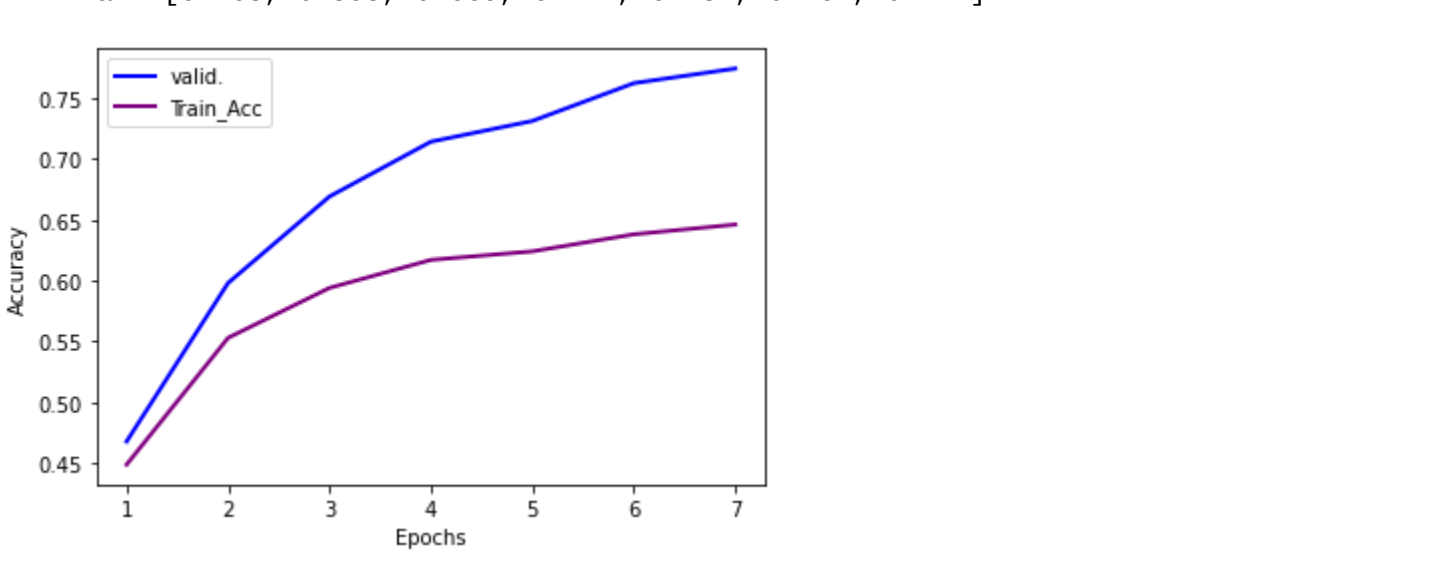
In [137]:
```python
Adam_Units_120_Cont_Yes()
```

```
****************************************************************************************************
Description
   Units:120
   Optimizer:Adam
   Content:Available

Epoch_time: 11.92 min
Total_time: 83.45 min
Loss:[2.552 1.189 0.884 0.745 0.658 0.606 0.555]
--------------------------------

ACCURACY: TRAIN-VALIDATION
    Val   [0.432, 0.666, 0.712, 0.738, 0.749, 0.754, 0.764]
    Train [0.454, 0.694, 0.76, 0.799, 0.816, 0.832, 0.846]
```



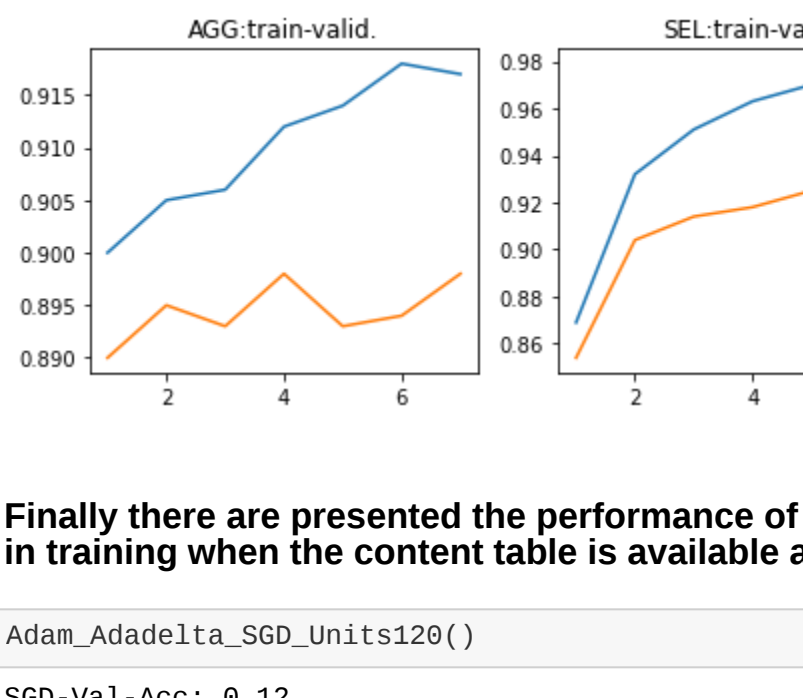


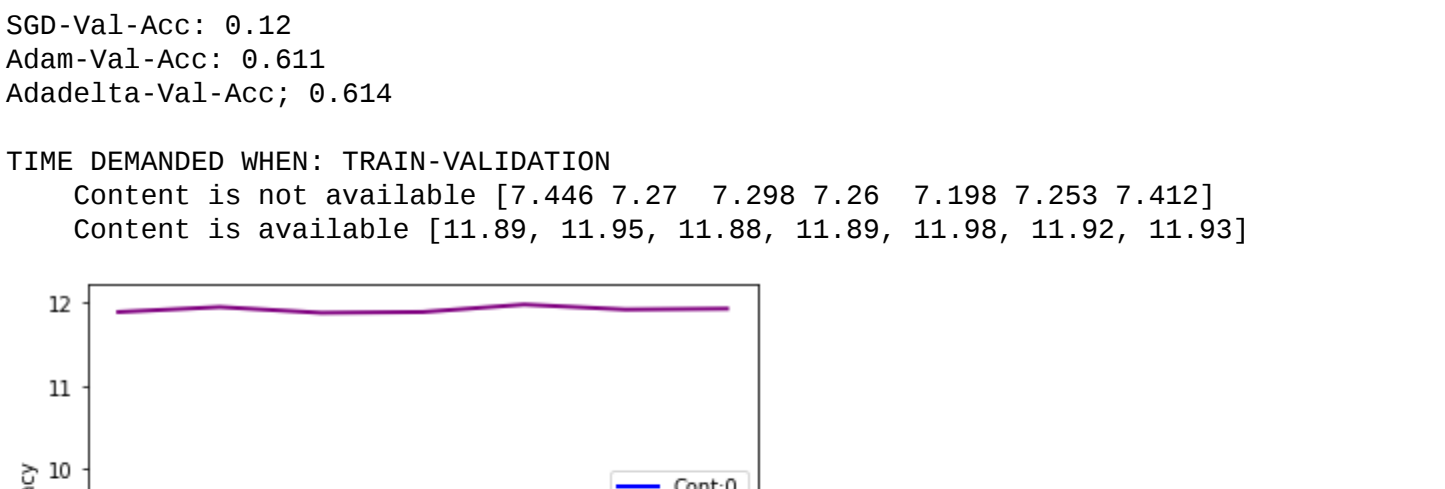
In [71]:
```python
Adam_Units_180_Cont_No()
```

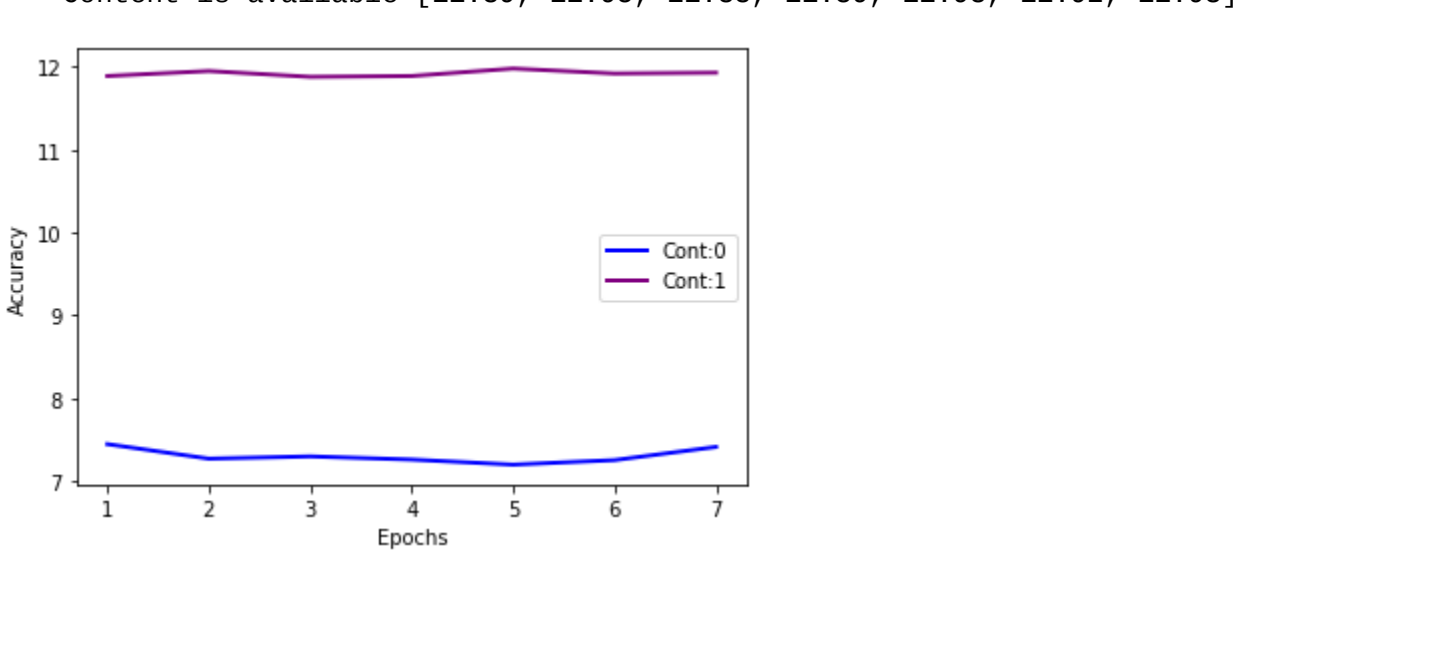
```
****************************************************************************************************
Description
   Units:180
   Optimizer:Adam
   Content:Not-Available

Epoch_time: 7.305 min
Total_time: 51.13 min
Loss:[2.735 1.313 1.016 0.858 0.765 0.673 0.62 ]
--------------------------------

ACCURACY: TRAIN-VALIDATION
    Val   [0.449, 0.553, 0.594, 0.617, 0.624, 0.638, 0.646]
    Train [0.468, 0.598, 0.669, 0.714, 0.731, 0.762, 0.774]
```





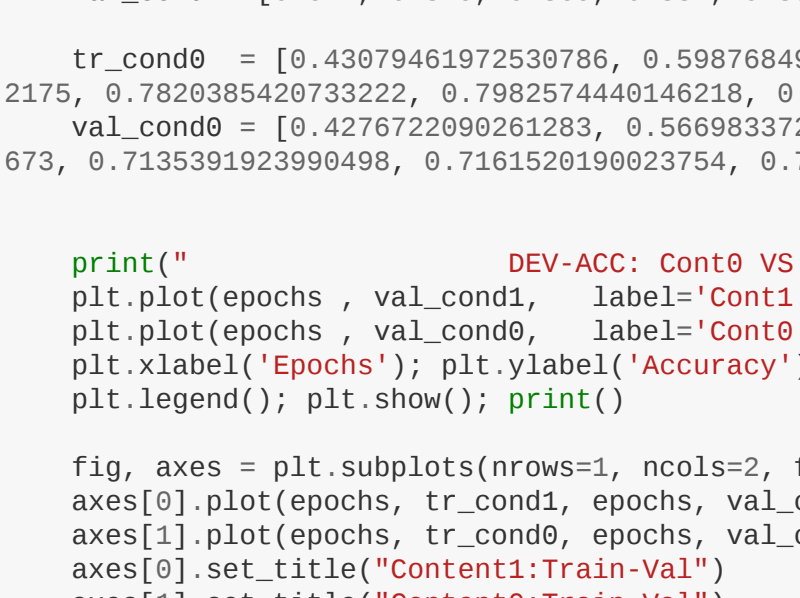


Finally there are presented the performance of different optimizer and the time difference in training when the content table is available and when not.

In [67]:
```python
Adam_Adadelta_SGD_Units120()
```

```
SGD-Val-Acc: 0.12
Adam-Val-Acc: 0.611
Adadelta-Val-Acc; 0.614

TIME DEMANDED WHEN: TRAIN-VALIDATION
    Content is not available [7.446 7.27  7.298 7.26  7.198 7.253 7.412]
    Content is available [11.89, 11.95, 11.88, 11.89, 11.98, 11.92, 11.93]
```



Plot Methods

In [70]:
```python
def Adam_120_Cont0_Cont1():
    print('**************************************************************************************************')
    print("Description Cont:0 vs Cont:1")
    print("   Units:120 "); print("   Optimizer:Adam");print()
    print("   Cont0: Avg_time=7.37 min ");
    print("   Cont1: Avg_time=11.92 min ");
    print('--------------------------------\n')

    # Train Validation
    epochs=[i+1 for i in range(EPOCHS)]
    tr_cond1=  [0.626, 0.838, 0.889, 0.917, 0.931, 0.937, 0.944]
    val_cond1= [0.614, 0.829, 0.869, 0.887, 0.896, 0.905, 0.904]

    tr_cond0  = [0.43079461972530786, 0.5987684991304965, 0.6960641658089931, 0.7436206835362175, 0.7820385420733222, 0.7982574440146218, 0.8201369911630053]
    val_cond0 = [0.4276722090261283, 0.5669833729216152, 0.6533254156769597, 0.6868171021377673, 0.7135391923990498, 0.7161520190023754, 0.7339667458432304]


    print("                   DEV-ACC: Cont0 VS Cont1")
    plt.plot(epochs , val_cond1,   label='Cont1.',    color='blue', linewidth=2)
    plt.plot(epochs , val_cond0,   label='Cont0', color='purple', linewidth=2)
    plt.xlabel('Epochs'); plt.ylabel('Accuracy')
    plt.legend(); plt.show(); print()

    fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 3))
    axes[0].plot(epochs, tr_cond1, epochs, val_cond1 )
    axes[1].plot(epochs, tr_cond0, epochs, val_cond0)
    axes[0].set_title("Content1:Train-Val")
    axes[1].set_title("Content0:Train-Val")
    fig.tight_layout(); plt.show()


def TIme_Cont1_Cont0():

    epochs=[i+1 for i in range(EPOCHS)]
    Loss0=[2.735, 1.313, 1.016, 0.858, 0.765, 0.673, 0.62 ]
    Loss1=[2.552, 1.189, 0.884, 0.745, 0.658, 0.606, 0.555]

    Time0=[7.44615, 7.270483333333333, 7.29795, 7.25951666666, 7.19841666666, 7.253133333333333, 7.41248333333]
    Time1=[np.round(x/60,2) for x in [713.274, 716.816, 712.957 ,713.583 ,718.938, 715.483 , 716.059]]


    print("Cont0-AVG-time: 7.30")
    print("Cont1-AVG-time: 11.92")
    print("Cont0-Loss:[2.735, 1.313, 1.016, 0.858, 0.765, 0.673, 0.62 ]")
    print("Cont1-Loss:[2.552, 1.189, 0.884, 0.745, 0.658, 0.606, 0.555]")

    fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 3))
    axes[0].plot(epochs, Loss0 , epochs, Loss1 )
    axes[1].plot(epochs, Time0, epochs, Time1  )
    axes[0].set_title("Loss: Cont0 vs Cont1")
    axes[1].set_title("Time per epoch: Cont0 vs Cont1")
    fig.tight_layout(); plt.show()

def Adam_Units_120_Cont_Yes():
    epochs=[i+1 for i in range(EPOCHS)]

    print('**************************************************************************************************')
    print("Description")
    print("   Units:120 "); print("   Optimizer:Adam"); print("   Content:Available");print()
    print("Epoch_time: 11.92 min ");
    print("Total_time: 83.45 min" );
    print("Loss:[2.552 1.189 0.884 0.745 0.658 0.606 0.555]")
    print('--------------------------------\n')

    # Train Validation
    Train_Acc= [0.454, 0.694,  0.76 ,  0.799, 0.816, 0.832, 0.846]
    Val_Acc=   [0.432, 0.666,  0.712,  0.738, 0.749, 0.754, 0.764]

    tr_agg=  [0.9  , 0.904, 0.906, 0.912, 0.908,  0.913, 0.918]
    tr_sel=  [0.76 , 0.891, 0.93 , 0.947, 0.958,  0.966, 0.972]
    tr_cond= [0.626, 0.838, 0.889, 0.917, 0.931,  0.937, 0.944]
    val_agg=  [0.895, 0.897,  0.898,  0.901,  0.895,  0.897, 0.9  ]
    val_sel=  [0.743, 0.87 ,  0.896,  0.904,  0.917,  0.915, 0.925]
    val_cond= [0.614, 0.829,  0.869,  0.887,  0.896,  0.905, 0.904]
    val_num=  [0.984, 0.992,  0.987,  0.992,  0.993,  0.993, 0.991]
    val_col=  [0.663, 0.871,  0.915,  0.926,  0.938,  0.944, 0.945]
    val_opr = [0.987, 0.983,  0.982,  0.983,  0.981,  0.982, 0.981]
    val_val=  [0.98 , 0.985,  0.985,  0.986,  0.985,  0.986, 0.986]



    print("ACCURACY: TRAIN-VALIDATION")
    print("    Val  ", Val_Acc); print("    Train",Train_Acc);
    plt.plot(epochs , Train_Acc,   label='valid.',    color='blue', linewidth=2)
    plt.plot(epochs , Val_Acc,label='Val_Acc', color='purple', linewidth=2)
    plt.xlabel('Epochs'); plt.ylabel('Accuracy')
    plt.legend(); plt.show(); print()

    fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 3))
    axes[0].plot(epochs, val_cond )
    axes[1].plot(epochs, val_num, epochs, val_col, epochs, val_opr, epochs, val_val)
    axes[0].set_title("Cond:Total (valid-set)")
    axes[1].set_title("Cond:Num-Col-Op-Value (valid-set)")
    fig.tight_layout(); plt.show()

    print()
    fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(10, 3))
    axes[0].plot(epochs, tr_agg, epochs, val_agg)
    axes[1].plot(epochs, tr_sel, epochs, val_sel)
    axes[2].plot(epochs, tr_cond, epochs, val_cond)

    axes[0].set_title("AGG:train-valid.")
    axes[1].set_title("SEL:train-valid.")
    axes[2].set_title("Cond:train-valid.")
    fig.tight_layout(); plt.show()



def Adam_Units_180_Cont_No():
    epochs=[i+1 for i in range(EPOCHS)]

    print('**************************************************************************************************')
    print("Description")
    print("   Units:180 "); print("   Optimizer:Adam"); print("   Content:Not-Available");print()
    print("Epoch_time: 7.305 min ");
    print("Total_time: 51.13 min" );
    print("Loss:[2.735 1.313 1.016 0.858 0.765 0.673 0.62 ]")
    print('--------------------------------\n')

    # Train Validation
    Train_Acc= [0.468, 0.598, 0.669, 0.714, 0.731, 0.762, 0.774]
    Val_Acc=   [0.449, 0.553, 0.594, 0.617, 0.624, 0.638, 0.646]

    tr_agg=  [0.9   ,0.905 , 0.906 ,  0.912 ,0.914 ,0.918 ,0.917];
    tr_sel=  [0.869 ,0.932 , 0.951 ,  0.963 ,0.97  ,0.975 ,0.979];
    tr_cond= [0.573 ,0.693 , 0.764 ,  0.805 ,0.819 ,0.846 ,0.859];
    val_agg= [0.89 , 0.895,  0.893,   0.898, 0.893, 0.894, 0.898]
    val_sel= [0.854, 0.904,  0.914,   0.918, 0.925, 0.921, 0.925]
    val_cond=[0.563, 0.658,  0.706,   0.727, 0.735, 0.751, 0.756]
    val_num= [0.949, 0.959,  0.96 ,   0.962, 0.961, 0.966, 0.968]
    val_col= [0.761, 0.851,  0.873,   0.88 , 0.884, 0.893, 0.895]
    val_opr= [0.962, 0.965,  0.969,   0.977, 0.978, 0.979, 0.978]
    val_val= [0.892, 0.883,  0.904,   0.907, 0.912, 0.913, 0.915]


    print("ACCURACY: TRAIN-VALIDATION")
    print("    Val  ", Val_Acc); print("    Train",Train_Acc);
    plt.plot(epochs , Train_Acc,   label='valid.',    color='blue', linewidth=2)
    plt.plot(epochs , Val_Acc,label='Train_Acc', color='purple', linewidth=2)
    plt.xlabel('Epochs'); plt.ylabel('Accuracy')
    plt.legend(); plt.show(); print()

    fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 3))
    axes[0].plot(epochs, val_cond )
    axes[1].plot(epochs, val_num, epochs, val_col, epochs, val_opr, epochs, val_val)
    axes[0].set_title("Cond:Total (valid-set)")
    axes[1].set_title("Cond:Num-Col-Op-Value (valid-set)")
    fig.tight_layout(); plt.show()

    print()
    fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(10, 3))
    axes[0].plot(epochs, tr_agg, epochs, val_agg)
    axes[1].plot(epochs, tr_sel, epochs, val_sel)
    axes[2].plot(epochs, tr_cond, epochs, val_cond)

    axes[0].set_title("AGG:train-valid.")
    axes[1].set_title("SEL:train-valid.")
    axes[2].set_title("Cond:train-valid.")
    fig.tight_layout(); plt.show()


def Seq2SQL_Where_Acc():

    epochs=[i+1 for i in range(EPOCHS)]
    dev_acc  =[0.154, 0.247, 0.300, 0.357, 0.387, 0.415, 0.444 ]
    train_acc=[0.165, 0.287, 0.359, 0.421, 0.458, 0.497, 0.527]



    print("Description")
    print("Epoch_time: 9.65 min ");
    print("Loss:[3.541 2.305 1.785 1.522 1.353 1.235 1.14]")
    print('--------------------------------\n')


    print("Seq2SQL-Where: TRAIN-VALIDATION")
    print("    Dev  ", dev_acc); print("    train",train_acc);
    plt.plot(epochs , train_acc,label='train', color='purple', linewidth=2)
    plt.plot(epochs , dev_acc,   label='dev.',    color='blue', linewidth=2)
    plt.xlabel('Epochs'); plt.ylabel('Accuracy')
    plt.legend(); plt.show(); print()


def Seq2SQL_Where_Acc():

    epochs=[i+1 for i in range(EPOCHS)]
    dev_acc  =[0.154, 0.247, 0.300, 0.357, 0.387, 0.415, 0.444 ]
    train_acc=[0.165, 0.287, 0.359, 0.421, 0.458, 0.497, 0.527]

    print("Description: Seq2SQL-Where TRAIN-VALIDATION")
    print("    Epoch_time: 9.65 min ");
    print("    Loss:[3.541 2.305 1.785 1.522 1.353 1.235 1.14]")
    print("    Dev  ", dev_acc); print("    train",train_acc); \
    print('--------------------------------\n')

    plt.plot(epochs , train_acc,label='train', color='purple', linewidth=2)
    plt.plot(epochs , dev_acc,   label='dev.',    color='blue', linewidth=2)
    plt.xlabel('Epochs'); plt.ylabel('Accuracy')
    plt.legend(); plt.show(); print()


def Adam_Adadelta_SGD_Units120():
    epochs=[i+1 for i in range(EPOCHS)]

    adam=[3.0792976994813492, 1.6387260944580901, 1.2547338484880088, 1.0440117774200257, 0.912056648367812, 0.8139558422196117, 0.7317942060046561]

    print("SGD-Val-Acc: 0.12")
    print("Adam-Val-Acc: 0.611")
    print("Adadelta-Val-Acc; 0.614")
    cont1=[np.round(x/60,2) for x in [713.274, 716.816, 712.957 ,713.583 ,718.938, 715.483 , 716.059]]
    cont0=[7.44615, 7.270483333333333, 7.29795, 7.259516666666667, 7.198416666666666, 7.253133333333333, 7.412483333333333]
    print()

    print("TIME DEMANDED WHEN: TRAIN-VALIDATION")
    print("    Content is not available", np.round(cont0,3)); print("    Content is available", cont1);
    plt.plot(epochs , cont0,   label='Cont:0',    color='blue', linewidth=2)
    plt.plot(epochs , cont1,label='Cont:1', color='purple', linewidth=2)
    plt.xlabel('Epochs'); plt.ylabel('Accuracy')
    plt.legend(); plt.show(); print()


def Units_60_120_180_ContNo():
    epochs=[i+1 for i in range(EPOCHS)]

    print('**************************************************************************************************')
    print("Description")
    print("   Optimizer:Adam"); print("   Content:Not-Available");print()
    print('--------------------------------\n')


    Train_Loss_60= [3.497, 1.969, 1.549, 1.325, 1.176, 1.068, 0.984]
    Train_Acc_60=  [0.195, 0.398, 0.48 , 0.535, 0.578, 0.61 , 0.642]
    Val_Acc_60=    [0.189, 0.373, 0.445, 0.476, 0.506, 0.531, 0.553]

    Train_Loss_180= [2.735, 1.313, 1.016, 0.858, 0.765, 0.673, 0.62 ]
    Train_Acc_180=  [0.468, 0.598, 0.669, 0.714, 0.731, 0.762, 0.774]
    Val_Acc_180=    [0.449, 0.553, 0.594, 0.617, 0.624, 0.638, 0.646]

    Train_Loss_120= [2.988, 1.487, 1.144, 0.964, 0.843, 0.769, 0.731]
    Train_Acc_120 = [0.379, 0.552, 0.626, 0.674, 0.706, 0.727, 0.742]
    Val_Acc_120  =  [0.361, 0.516, 0.567, 0.589, 0.607, 0.618, 0.618]

    print("Loss: 60-120-180")

    plt.plot(epochs , Train_Loss_60 ,   label='units:60.',    color='blue', linewidth=2)
    plt.plot(epochs , Train_Loss_120,   label='units:120',    color='blue', linewidth=2)
    plt.plot(epochs , Train_Loss_180,   label='units:180.',    color='green', linewidth=2)
    plt.xlabel('Epochs'); plt.ylabel('Loss')
    plt.legend(); plt.show(); print()


    print("    valAcc_60:", Val_Acc_60); print("    valAcc_120",Val_Acc_120);  print("    va
```